NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

USE OF MULTIPLE TRACKING DATA IN THE
CALIBRATION OF SHORT BASELINE ARRAYS

by

Joseph A. Gembarski

March 1992

Thesis Advisor: Robert R. Read

Approved for public release; distribution is unlimited.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL OR	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. | PROJECT NO. | TASK NO. | WORK UNIT ACCESSION NO.

11. TITLE (Including Security Classification)
USE OF MULTIPLE TRACKING DATA IN THE CALIBARATION OF SHORT BASELINE ARRAYS

12 PERSONAL AUTHOR(S)
GEMBARSKI, Joseph A.

13 TYPE OF REPORT Master's thesis	13b. TIME COVERED FROM TO	14. DATE OF REPORT (Year, Month, Day) 1992, March	15. Page Count 151

16. SUPPLEMENTAL NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Consistency of errors, calibration of test ranges, three dimensional position location systems, simulate annealing

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The calibration problem of a Short Baseline Underwater Vehicle Tracking Range is concerned largely with the coherency of path as the target vehicle passes from the domain of one array into that of another. The arrays are placed approximately in a hexagonal mesh. Thus there are regular locations (triple overlap regions) where the vehicle is tracked simultaneously by three separate arrays. Longbase methods can be used to locate the vehicle in these isolated regions. Presumably the four determinations (three for the individual short baseline arrays and one for the long baseline) can be used to locate the arrays and help calibrate the range.

This thesis contains a feasibility study for this idea. Vehicles are placed at known locations in the triple overlap regions. A sequence of correction actions is postulated and the algorithms have been programmed. It appears to work quite well within the confines of this idealized study.

20 DISTRIBUTION/AVAILABLTIY OF ABSTRACT X UNCLASSIFIED/UNLIMITED □ SAME AS RPT. □ DTIC	1a. REPORT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Robert R. Read	22b. TELEPHONE (Include Area Code) (408)646-2382	22c. OFFICE SYMBOL OR/Re

Previous editions are obselete.
S/N 0102-LF-014-6603
SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

Use of Multiple Tracking Data in the Calibration of Short Baseline Arrays

by

Joseph A. Gembarski
Lieutenant, United States Navy
B.S., Michigan State University, 1984

Submitted in partial fulfillment of the
requirements for the degree of

MASTERS OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 1992

Author:

Joseph A. Gembarski

Approved by:

Robert R. Read, Thesis Advisor

James N. Eagle, Second Reader

Peter Purdue, Chairman
Department of Operations Research

ABSTRACT

The calibration problem of a Short Baseline Underwater Vehicle Tracking Range is concerned largely with the coherency of path as the target vehicle passes from the domain of one array into that of another. The arrays are placed approximately in a hexagonal mesh. Thus there are regular locations (triple overlap regions) where the vehicle is tracked simultaneously by three separate arrays. Longbase methods can be used to locate the vehicle in these isolated regions. Presumably the four determinations (three for the individual short baseline arrays and one for the long baseline) can be used to locate the arrays and help calibrate the range.

This thesis contains a feasibility study for this idea. Vehicles are placed at known locations in the triple overlap regions. A sequence of correction actions is postulated and the algorithms have been programmed. It appears to work quite well within the confines of this idealized study.

THESIS DISCLAIMER

The reader is cautioned that the computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

DEDICATION

This thesis is dedicated to the following men and women:

To Jean, thanks for bringing the sun and stars to Monterey.

To all the people who live in a linear world, three points do form a volume, all curves are straight, and a diamond is naught but an ellipse.

To Howard Roark and John Galt - you have reaffirmed my knowledge in all - be true.

I. INTRODUCTION

The Naval Undersea Weapons Engineering Station (NUWES) currently operates nine test ranges for the purpose of testing and analyzing the performance of torpedoes. These ranges provide a three-dimensional track of torpedoes (or other water borne vehicles) from acoustic information received by bottom based sensor arrays. This thesis supports ongoing research conducted by Professor Robert R. Read of the U. S. Naval Postgraduate School, whose goal is to monitor the calibration of these ranges.

The focus of this thesis is to support range calibration, specifically, by investigating sensor location error (position and orientation) using raw tracking data provided by the range itself. Since the sensors are inaccessible on the ocean floor, their exact locations are in doubt and estimated by a range survey procedure. The survey is costly and takes an entire day for a single sensor array, requiring that the range be shutdown, thus losing valuable testing time. This OPERATIONAL estimate of sensor location is subject to error and may differ from its ACTUAL location. Successive resurveys of sensor arrays find the array locations moved by as much as tens of feet.

A difficulty in using the range, is that two or more versions of a torpedo track can be provided by adjacent sensing arrays. These differing versions of track manifest themselves as the torpedo traverses areas of mutual sensor monitoring (overlap regions). Accurate study of torpedo performance becomes difficult when considering the joint behavior of several vehicles (e.g. a submarine and torpedo).

The calibration support methodology developed in this paper will focus on treating the location of those sensing arrays that are interior to the others (see Figure 3). The results will be a set of decision rules for the iterative correction of the interior sensorꞅ location. Appendix I contains a mathematical formulation of the problem and explains the choice of algorithm made. In the context presented here, this methodology works quite well.

The organization of the thesis is as follows:
Chapter II contains explicit background of the range setup and operation along with a description of locational errors. Chapter III contains the error correction methodology developed to solve for sensor locational errors. Specific assumptions are included here for this problem. Chapter IV integrates other sensors into the correction methodology and relaxes the initial assumption made in the solution formulation. Chapter V reports on the conclusions reached and recommendations for further work in this area.

II. BACKGROUND

A. RANGE CONSTRUCTION AND OPERATION

References 1-6 outline previous work done in support of this project. This thesis, as with previous works, will utilize data from the Nanoose range.

The Nanoose underwater tracking range consists of a number of sensing arrays of the short baseline (SBL) type positioned on the ocean bottom. Each array is three dimensional and contains four hydrophones (see Figure 1).

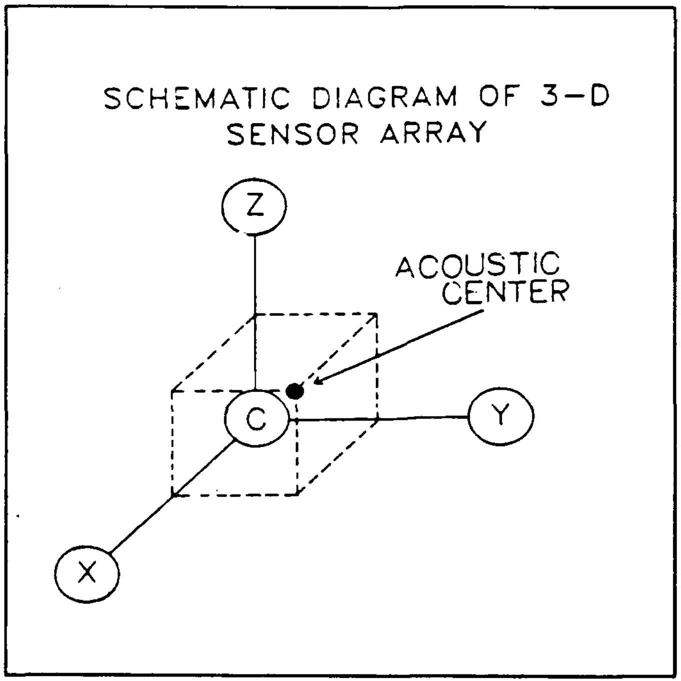

Figure 1. SBL Sensor Array.

The acoustic center is the geometric center of the box formed by the arms of the sensor array and is used as the sensor array's location. The 30 foot sensor arms are short in comparison to the target tracking distances, thus the name "short baseline" (SBL).

Targets on the range are tracked in the following manner. The targets (e.g. torpedoes) are fitted with acoustic transmitters (pingers) which emit sound pulses at precisely timed intervals. The pingers and sensor arrays, are synchronized with the shore based clock. When an array detects an acoustic pulse, the transit times of the sound wave, from the target to the sensor array, are computed.

A unique relative target position location (posloc) is found using the transit times measured from the target pinger to the four sensor hydrophones. The individual sound pulses are coded for the purpose of distinguishability. Figure 2 shows a sensor array interacting with various target vehicles.

A technique known as ray tracing is used to determine location. A ray is the normal to the acoustic wave front. Based on Snell's Law [Ref.7], ray tracing reconstructs the ray path traveled by sound in water. The differential of pulse arrival times at the four hydrophones are used to construct the ray's azimuth and elevation angles at the array's location. These are used to initialize the ray tracing algorithm. The computation works its way back to the sound source and the algorithm is stopped when the transit time is

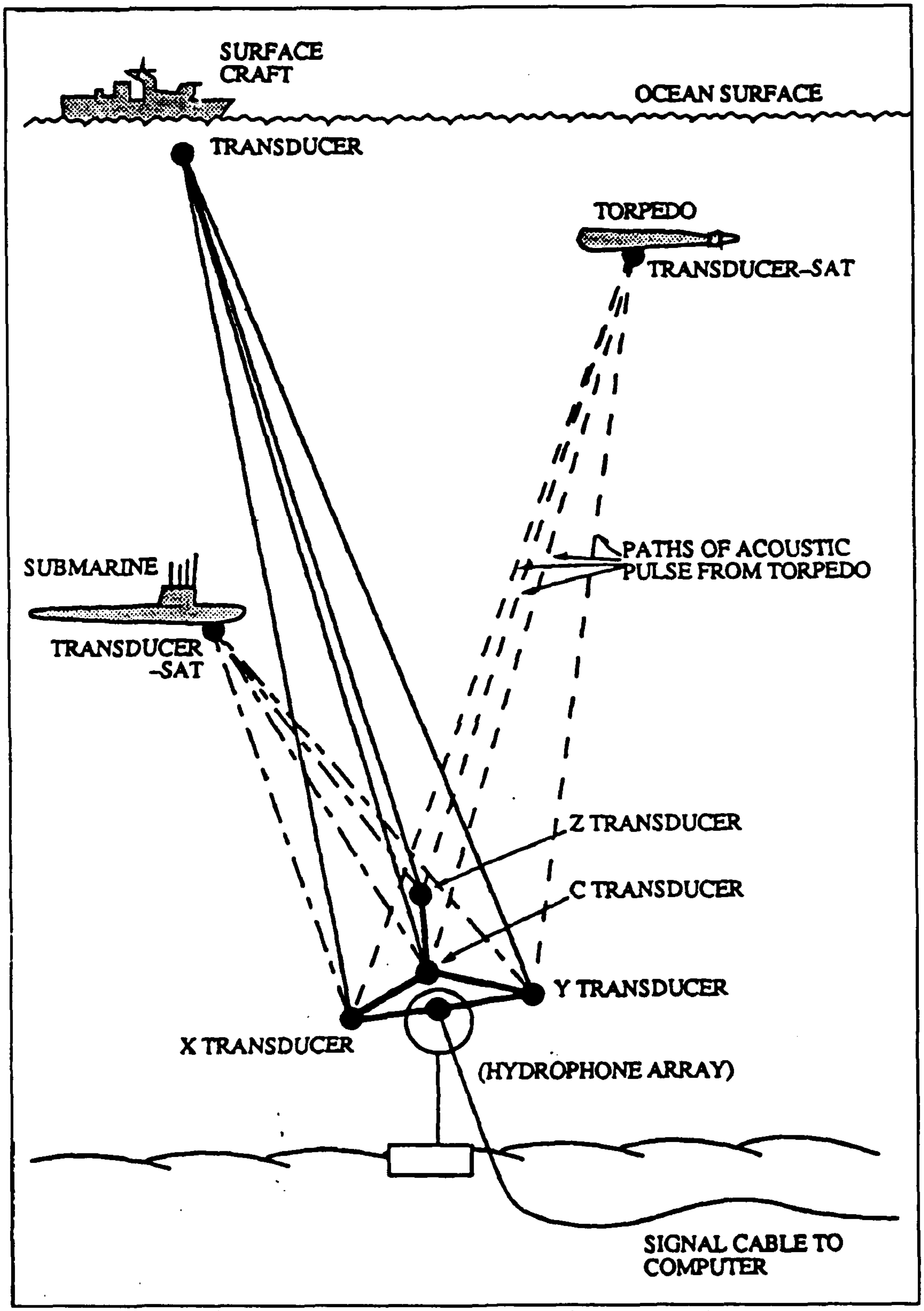

Figure 2. SBL Sensor and Ray Paths

'used up'. The result is a posloc for the target at the time of the ping.

Reconstruction of the target local track is accomplished by joining the SBL poslocs of each array over time. This produces a relative or apparent target track from each sensor. Connecting these relative tracks gives a target track throughout the range. The use of many sensor arrays (Figure 3), enables continuous monitoring of the entire range. Each array monitors a finite volume of water, limited by both acoustical attenuation and time gating. Careful arrangement of the sensor arrays enables full coverage without excessively sized overlap regions. These overlap regions are necessary to ensure track continuity throughout the range.

There are two types of overlap regions, double and triple. The names correspond to the numbers of arrays which serve each, two or three respectively. An example of a torpedo traversing a double overlap region is presented in Figure 4. This shows an ideally continuous, or coherently connected, track as the torpedo crosses from one sensor region to another.

B. POSLOC ERRORS

The track versions generated by the individual sensors are rarely connected as the target crosses these overlap regions. Separations in track are commonly on the order of 40-50 feet (slant range).

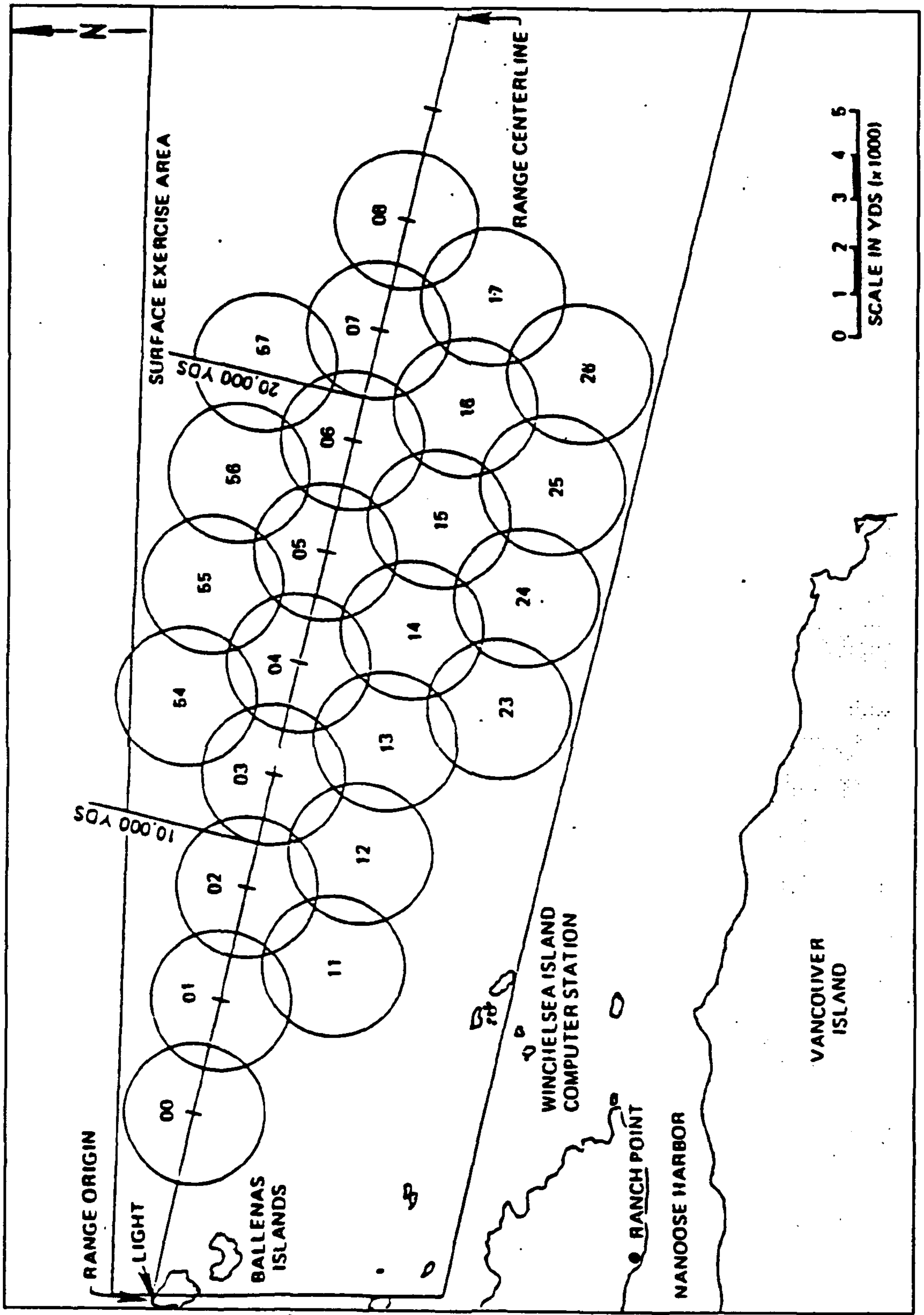

Figure 3. Nanoose Range.

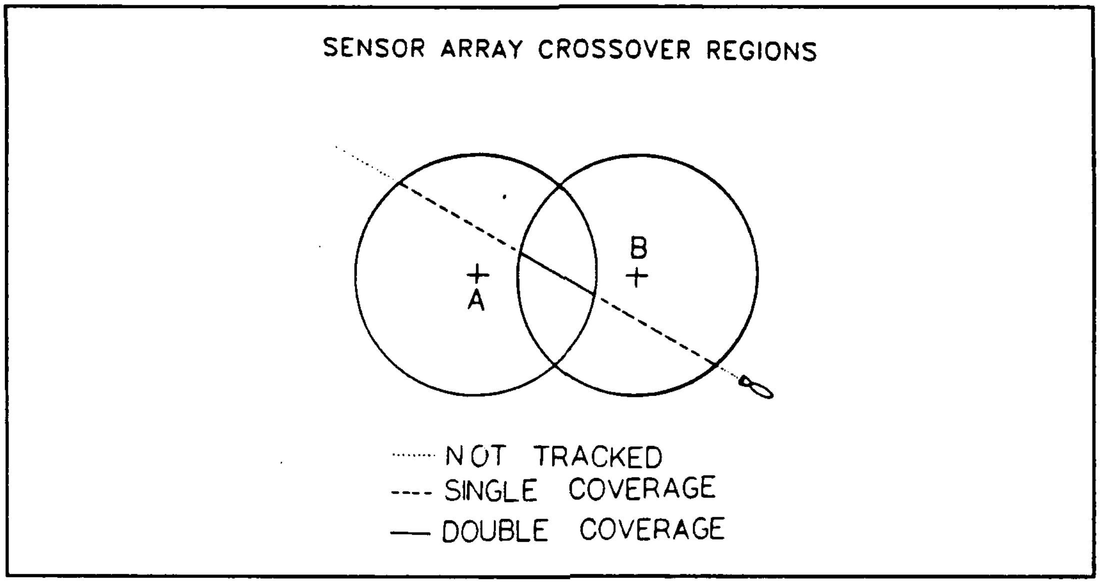

Figure 4. Torpedo Track Through a Double Overlap Region (no sensor displacement errors).

The causes of these discontinuities can be grouped into 3 general categories:

1) Water Condition Errors/Ray tracing

   Reference 1 contains an analysis of this problem.

2) Pinger Timing Errors (Offset and Synchronization)

   Reference 2 details work done in this area.

3) Sensor Location Errors (Position and Orientation)

   References 3-6 contain previous work done in this area. This thesis continues this work.

Examination of the current ray tracing initialization method, shows that it can contribute up to 1/3 of the total target positional errors. A recommended method of initialization presented in Appendix F, reference 1, significantly reduces the errors caused by the current ray trace initialization method.

The remaining significant errors are believed to be attributed to water column and sensor locational errors. In the water column, two effects come into play. First, the horizontal non homogeneity of the water causes the ray paths to become kinked. The second effect is the limited depth to which the depth velocity profile (DVP) is measured. The range obtains but one DVP measurement per day (due to cost and equipment constraints) and to a depth generally shallower than the deeper sensor hydrophones. Extrapolation of the DVP is required if a sensor hydrophone is deeper than the level provided by the measured profile. Uncorrected for, these effects can cause large posloc displacements.

The location errors of the sensors lend themselves to relatively simple analyses and are discussed below.

A number of other systematic error sources are present (e.g. bottom currents, curvature of the earth), but their effects are of much smaller magnitude. For the present they are ignored.

C. LOCATIONAL ERRORS

A sensor location comprises six components. These are the x, y, and z position of the sensors acoustic center (in the range coordinate system), and the xtilt, ytilt, and zrot angular orientation of each array axis from the range coordinate axis. The tilt angles are measured to the sensor's arms from the horizontal. The zrot component measures the CCW rotation of the sensor from east (zrot=0 implies the x-arm is pointed east). The xtilt, ytilt and zrot can be thought of in terms of roll, pitch and yaw , respectively. The x, y, z, and zrot terms are measured during the sensor survey. Such a survey locates the acoustical center of the array with respect to the range coordinate system. When this is combined with the tilt angles obtained from the sensor's x and y arms the array's OPERATIONAL location is determined.

The following terms are defined:

- <u>OPERATIONAL location</u> - the position and orientation of the sensor array as determined by survey techniques.
- <u>ACTUAL location</u> - the true position and orientation of the sensor array.

Surveyed positions will suffer from the category 1 errors, water column nonhomogeneity and ray tracing. Another possible cause for the deviations from previous OPERATIONAL locations (survey to survey) can be due to physical array shifts on the ocean floor. If such shifts actually do occur, previous locational data is invalidated. Thus range monitoring must be performed continuously.

The last two components of the sensors location, xtilt and ytilt, can differ from the ACTUAL tilts due to inaccuracies of the indicators in the arms of the sensors.

It is easy to visualize in the two dimensional case the effect on the posloc from errors in the six components of the sensor location. A situation similar to that depicted in Figure 5 occurs if we confine the location errors to just x, y and rotational offsets. This is attributed to a shift in the local coordinate system of the sensor from which the poslocs are found. Posloc errors are less direct for errors in the z direction. Not only is there a change in the depth of the local coordinate system, but also in the water environment. As the sensor is 'displaced' in depth, the sound velocity at the new level of the hydrophones will be

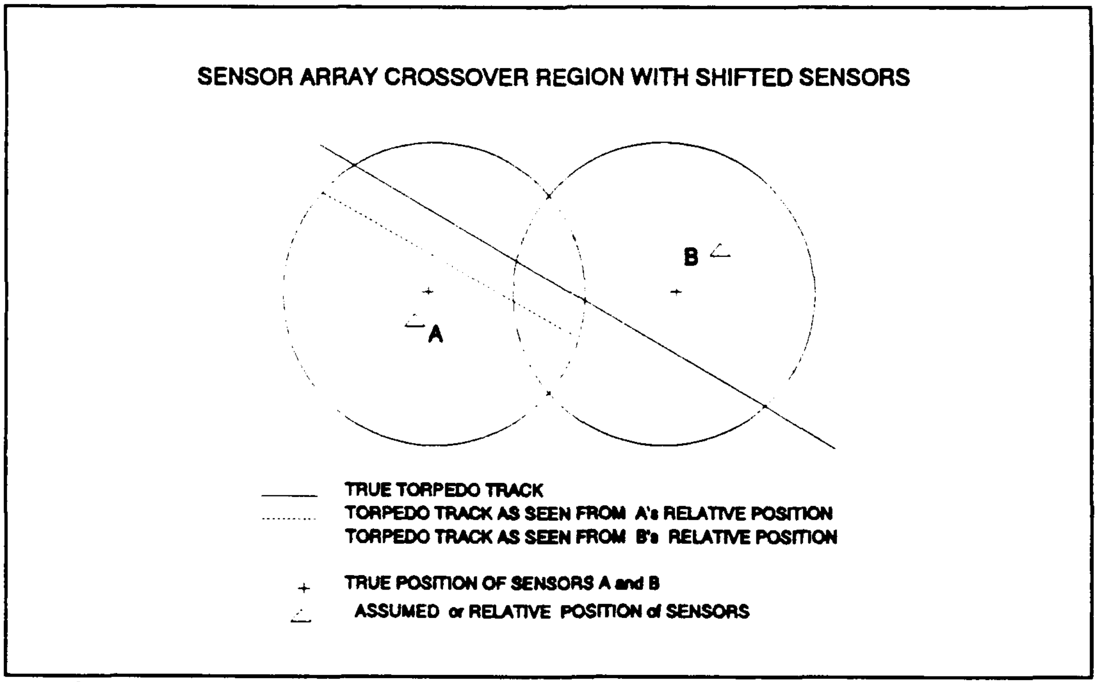
SENSOR ARRAY CROSSOVER REGION WITH SHIFTED SENSORS
B
A
TRUE TORPEDO TRACK
TORPEDO TRACK AS SEEN FROM A's RELATIVE POSITION
TORPEDO TRACK AS SEEN FROM B's RELATIVE POSITION
TRUE POSITION OF SENSORS A and B
ASSUMED or RELATIVE POSITION of SENSORS

Figure 5. True and observed Torpedo Track as seen by displaced sensors.

different. This changes the ray paths of the sound pulses and causes a disproportional change in the target posloc direction vector.

For an example of the size of the errors produced from deviations in the six locational components, consider a 0.5° tilt error (xtilt or ytilt). This will cause a SBL posloc displacement of 40 feet at a 4000 foot range. This is a typical value of interest in an overlap region.

III. METHODOLOGY

The adjustments to the OPERATIONAL location of SBL sensing arrays, will be first to the interior arrays. Interior arrays are those sensor arrays which have triple overlap with six adjacent, or satellite, sensor arrays (see Figure 3).

A. ERROR ANALYSIS

Figure 6 contains a schematic diagram depicting the triple overlap regions which provide information to the interior array. A triple overlap region (henceforth referred to as a region), represents a volume of water monitored, or served, by three independent sensor arrays; in the present case the specified interior array and two satellite arrays. Three independent poslocs will be generated for a given acoustic 'ping'. In actuality these three poslocs will rarely be coincident. Their determination will be influenced by the error sources mentioned previously.

1. Consensus Location (LONGBASE)

The SBL poslocs are evaluated upon comparison with a consensus location. The choices for this location are one, or both of the other satellite poslocs, or some aggregate location of the three individual poslocs. Professor Read developed an algorithm which, assuming that the sensor arrays that service a region are positionally correct (x, y, and z),

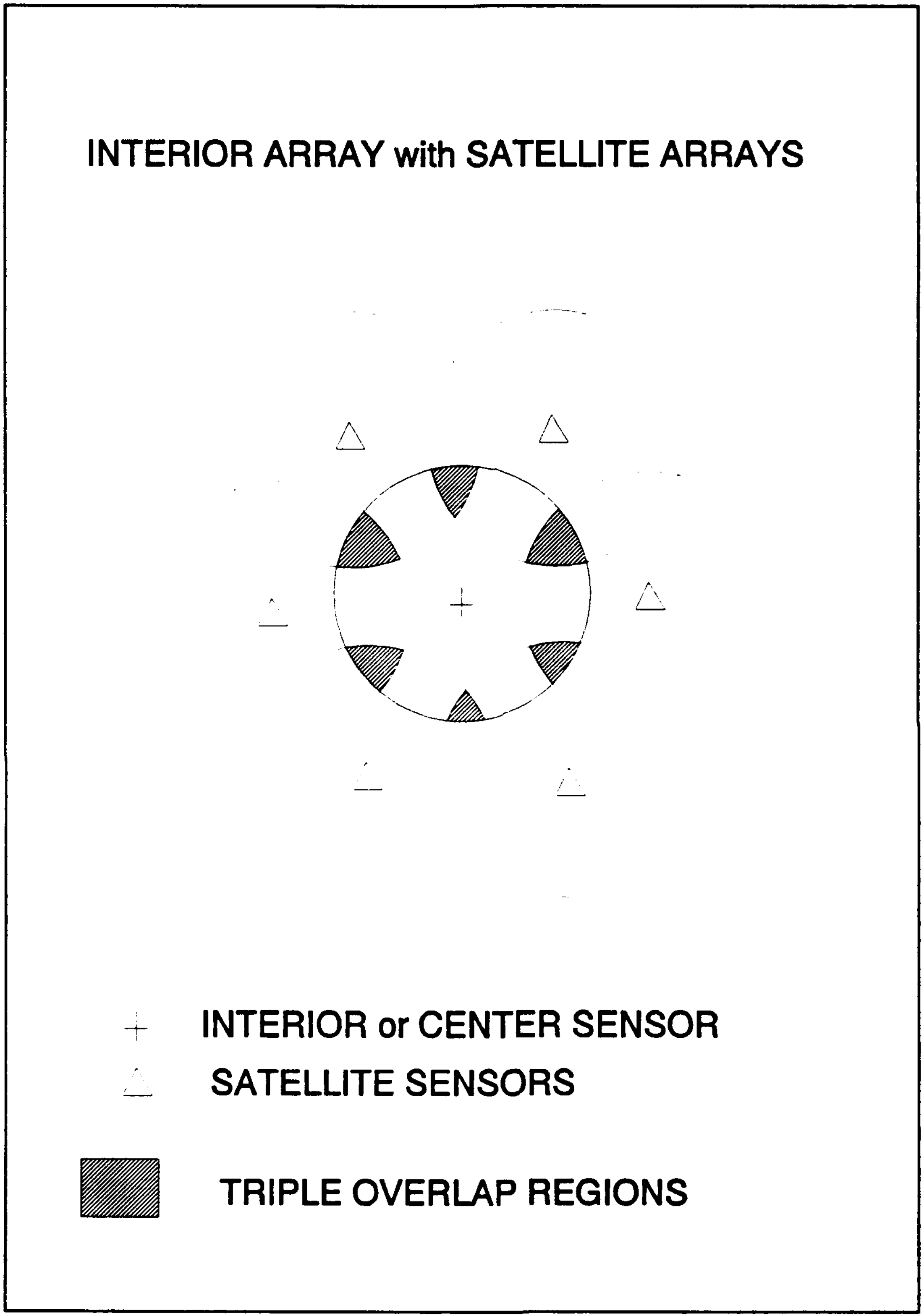
INTERIOR ARRAY with SATELLITE ARRAYS
+ INTERIOR or CENTER SENSOR
△ SATELLITE SENSORS
▨ TRIPLE OVERLAP REGIONS

Figure 6. Interior Array and its Triple Overlap Regions.

adjusts the azimuth and elevation angles (associated with each of the three arrays) so that the three poslocs coincide [Ref. 3]. This method, long baseline (LBL), uses the transit time information from the C- hydrophones of the three sensor arrays in a region. In this way a 'new coordinate' system from these hydrophones is formed. Since the 'arms' of this newly formed surrogate array are long in comparison to the distance to the target, this is termed "long baseline".

Initially the transit times from each array are converted into the three SBL poslocs, which are not coincident. Each posloc falls on the surface of a 'sphere' described by varying the azimuth and elevation angles for a fixed transit time. Each of the three arrays produces this 'spherical' surface and they intersect at a single point. This point may be located either upward or downward in depth, and perhaps at a different angular location than the SBL poslocs.

Since no sensor positional errors are assumed in longbase, a problem develops in that the depth of the LBL posloc may be 'forced' from the averaged depth of the SBL poslocs. This may or may not move it closer to the true depth. As the longbase algorithm determines the depth of this consensus point, the horizontal position is also updated to maintain constant transit times. This program is the main tool exploited in our developments. The exploitation of this program will be described subsequently.

2. Consistency Of Errors

A formulation of the problem and reasons for the choice of a coordinate descent algorithm are found in Appendix I. Once a consensus location for the target is found (LBL posloc), an error table can be made for the specified interior array. This table will consist of the differences between the LBL and SBL poslocs and contains the following locational attributes: elevation angle, azimuth angle, horizontal range, x, y, and z coordinates. Note that redundant information is supplied. The azimuth, elevation angles and horizontal range, can be converted to x, y and z coordinates.

To describe the exploitation of the LBL algorithm, consider an interior array which contains six regions. Suppose that the OPERATIONAL and ACTUAL location of the six satellite arrays are the same, but not so for the interior array. Then the poslocs produced by longbase will provide information about the location of the interior array. (It has been verified that for three coincident poslocs, the LBL posloc generated will fall on that coincident point.)

Figure 7 illustrates an interior array and two of its regions along with the LBL and SBL poslocs in two dimensions. Expanded views of the two regions are shown in the wings. One can perceive a correction to the right and upwards. Next consider six satellite regions. One can imagine a consistency in the offset and rotational nature of the x and y errors. Heuristically, the optimal location for the interior array can

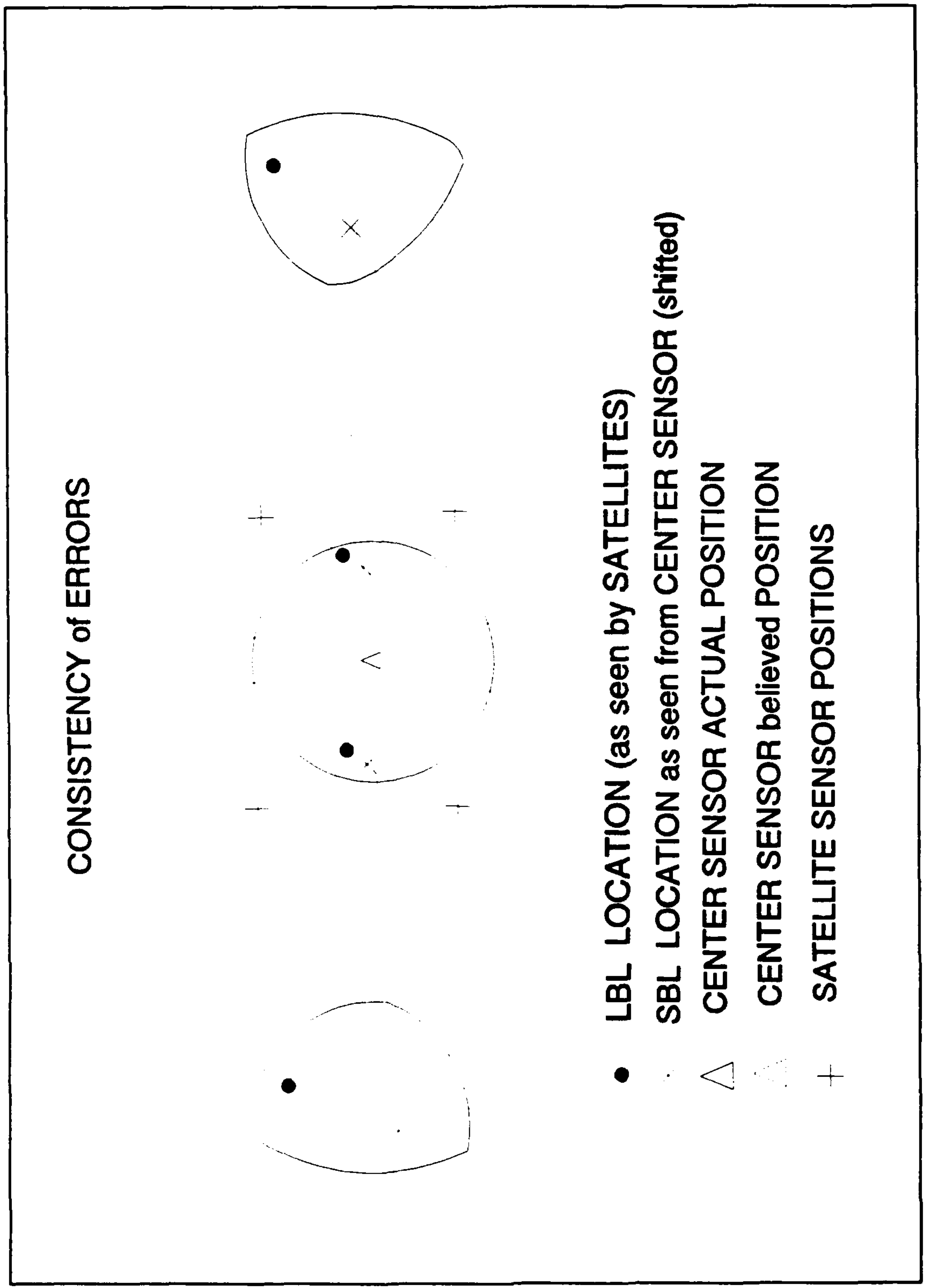
CONSISTENCY of ERRORS
● LBL LOCATION (as seen by SATELLITES)
· SBL LOCATION as seen from CENTER SENSOR (shifted)
◁ CENTER SENSOR ACTUAL POSITION
◁ CENTER SENSOR believed POSITION
+ SATELLITE SENSOR POSITIONS

Figure 7. Consistency of Errors for Interior Array in Two Triple Overlap Regions.

be found by correcting one 'error' at a time.

B. ASSUMPTIONS

The following assumptions are made in order to develop a prioritized set of corrections for the selected interior sensing array.

1) SATELLITE SENSORS ARE ERROR FREE

This assumption will allow full concentration on the posloc errors produced by the center array, without interference from noise in the locations of the satellite arrays.

2) HOMOGENEOUS ISOSPEED WATER COLUMN (DVP)

The advantage to making this assumption is that of simplification and the elimination of the errors produced in generating the SBL poslocs using the current algorithms. This simplifies the development of the error correction priorities. Posloc positional errors are now isolated to sensor array locational errors.

C. TERMINOLOGY

The following terminology is used throughout the paper and in the FORTRAN programs.

For sensor number k, and overlap region j (j=1 to 6):

$\{X_{k,i}\}$ position of the acoustic center of sensor array k, for i=1-3 (x, y, and z) in the Range coordinates (depth positive downward)

$\{A_{k,i}\}$ angular orientation of sensor array k for i= 1-3 (xtilt, ytilt, and zrot). xtilt and ytilt measured from the horizontal, zrot measured CCW from the Range centerline

$\{ERROR_{k,i}\}$ induced (known) locational error to sensor array k, for i=1-6 (x, y, z, xtilt, ytilt, and zrot)

$\{SHIFT_{k,i}\}$ corrections made to sensor array k's location for i=1-6 (x, y, z, xtilt, ytilt, and zrot)

$\{TIME_{k,j,i}\}$ acoustic ping transit time from the target to sensor k's hydrophones i=1-4 (x, y, z, and c) in region j

$\{XS_{k,j,i}\}$ SBL posloc position for sensor k in region j for i=1-3 (x, y, and z) in the Range coordinates

$\{XL_{k,j,i}\}$ LBL posloc position for sensor k in region j for i=1-3 (x, y, and z) in the Range coordinates

$\{PHS_{k,j}\}$ azimuth angle (CCW from east) to sensor k's SBL posloc in region j from the c hydrophone

$\{PHL_{k,j}\}$ azimuth angle (CCW from east) to sensor k's LBL posloc in region j from the c hydrophone

$\{THS_{k,j}\}$ elevation angle, from horizontal, to sensor k's SBL posloc in region j from the c hydrophone

$\{THL_{k,j}\}$ elevation angle, from horizontal, to sensor k's LBL posloc in region j from the c hydrophone

$\{HS_{k,j}\}$ horizontal range from sensor k's c hydrophone to region j's SBL posloc

$\{HL_{k,j}\}$ horizontal range from sensor k's c hydrophone to region j's LBL posloc

Utilizing the above definitions, we have for sensor array k:

ACTUAL LOCATION: $\{X_{k,1} \quad X_{k,2} \quad X_{k,3} \quad A_{k,1} \quad A_{k,2} \quad A_{k,3}\}$

OPERATIONAL LOCATION: $\{(X_{k,1}+\mathrm{ERROR}_{k,1})\ (X_{k,2}+\mathrm{ERROR}_{k,2})\ (X_{k,3}+\mathrm{ERROR}_{k,3})$
$(A_{k,1}+\mathrm{ERROR}_{k,4})\ (A_{k,2}+\mathrm{ERROR}_{k,5})\ (A_{k,3}+\mathrm{ERROR}_{k,6})\}$

Solving for $\{\mathrm{SHIFT}_{k,i}\}$, the ACTUAL location of the sensor array will be obtained as such:

$$\begin{aligned}
X_{k,1} &= (X_{k,1}+\mathrm{ERROR}_{k,1}) + \mathrm{SHIFT}_{k,1}\\
X_{k,2} &= (X_{k,2}+\mathrm{ERROR}_{k,2}) + \mathrm{SHIFT}_{k,2}\\
X_{k,3} &= (X_{k,3}+\mathrm{ERROR}_{k,3}) + \mathrm{SHIFT}_{k,3}\\
A_{k,1} &= (A_{k,1}+\mathrm{ERROR}_{k,4}) + \mathrm{SHIFT}_{k,4}\\
A_{k,2} &= (A_{k,2}+\mathrm{ERROR}_{k,5}) + \mathrm{SHIFT}_{k,5}\\
A_{k,3} &= (A_{k,3}+\mathrm{ERROR}_{k,6}) + \mathrm{SHIFT}_{k,6}
\end{aligned}$$

D. SETUP OF TEST CONDITIONS (SIMSTRT, SETUP)

First values of the $\{\mathrm{ERROR}_{k,i}\}$ vector defined above are prescribed for each sensor array k, in order to generate the OPERATIONAL locations. Next the DVP to be used is selected. This DVP will be 4880 ft/sec and extend to a depth exceeding the deepest hydrophone in order to eliminate the need for DVP extrapolation. The starting points for our 'pseudo target' are arbitrarily chosen to be in the center of each triple overlap region j, at a depth of 400 ft. The transit times $\{\mathrm{TIME}_{k,j,i}\}$, i=1-4, of acoustic pings are then generated from these starting points to the ACTUAL array locations. The are times derived using the RAYFIT program [Ref. 1], are analogous to the times provided by the range.

With the transit times and the OPERATIONAL location of the sensor arrays, SBL poslocs are generated through ray tracing. These SBL poslocs are the sensors perceived target position. The three poslocs in each region are then used to generate the

LBL poslocs. An error table for each sensor array is then formed using the LBL and SBL poslocs. This table contains the following LBL-SBL posloc difference information for sensor k and region j:

$THER_{k,j} = THL_{k,j} - THS_{k,j}$, elevation angle differences
$PHER_{k,j} = PHL_{k,j} - PHS_{k,j}$, azimuth angle differences
$HER_{k,j} = HL_{k,j} - HS_{k,j}$, horizontal range differences
$XER_{k,j} = XL_{k,j,1} - XS_{k,j,1}$, x coordinate differences
$YER_{k,j} = XL_{k,j,2} - XS_{k,j,2}$, y coordinate differences
$ZER_{k,j} = XL_{k,j,3} - XS_{k,j,3}$, z coordinate (depth) differences.

E. ERROR FIXING (ERRFIX)

1. Measure of Error

The goal is to minimize separation between the LBL poslocs and the interior arrays SBL poslocs in each region. A measure of error to gauge success is the three dimensional distance between the two poslocs:

$$d = \sqrt{XER^2+YER^2+ZER^2}.$$

Instead of this we will use an approximate equivalent measure developed by Professor Read in reference 1:

$$d = \sqrt{ZER^2+HER^2+(HL\times PHER)^2}.$$

This measure of error contains the horizontal separation distance ($HER^2 = XER^2 + YER^2$), and an additional term to account for azimuthal errors. It's use circumvents the need for transforming spherical coordinates to cartesian coordinates.

The total measure of error for a sensor array, will be the above measure averaged over the sensor's regions.

2. Individual Error Corrections

Error corrections can be thought of in four distinct phases: Azimuth, Depth (Z) by Elevation Angle, Tilt Angles (xtilt and ytilt), and coordinates (X, Y, and Z). This section examines individual locational error corrections. These individual error components are treated in isolation of the other locational errors that may exist for the sensor array. The next section will look at how to combine these error correction components into a integrated correction scheme to solve for array location.

The component error corrections will be based on the average value of the individual errors of interest. The use of averaged error allows the optimal sensor location to be approached without excessive correction overshoot.

Although an isospeed DVP is used, the algorithms will be written generally so as to facilitate the transition into a gradient environment. A correction scheme for each error independent of the others will be looked at first. A method to combine these into a correction algorithm will then be introduced.

a. *Azimuth Errors*

An azimuth error (rotational error), of the interior array will manifest itself in each region regardless of the water environment. Due to the ranges to the triple overlap regions, small azimuth errors will cause large posloc separations. Since these errors are easy to spot and their correction greatly minimizes posloc separation, an attempt will be made to correct these first.

The errors due to azimuth will be of similar magnitudes in each region, but most importantly, of the same (consistent) sign. The magnitude of these errors in each region can be masked by even moderate x and y coordinate displacements of the interior sensor. In correcting for this azimuth error the average azimuth error over the regions is used as an initial guess. This first guess will generally be smaller in magnitude than the true azimuth error.

b. *Depth by Elevation Angle*

Once again consistency in the signs of the elevation angles is sought. Elevation angles of the same sign can be caused by a depth error to the interior array. As above for the azimuth errors, a depth correction can be made if the elevation angles possess the same sign. The magnitude of the correction will again be based on the average of the depth errors. Correction of this angular error using depth, vice a function of the elevation angles was chosen for simplicity. Correcting for depth errors in this way also has the coincident benefit of correcting for elevation angle errors.

In the isospeed environment, longbase is biased toward the error free (correct) satellite poslocs. A depth error to the interior array also correlates directly to a depth shift in its SBL posloc. This results because the SBL posloc direction vector is not effected by the depth change in the isospeed environment. Thus a positive, consistent, elevation error is induced by an OPERATIONAL depth that is deeper than ACTUAL.

Transitioning to a gradient environment, longbase still remains biased toward the 'correct' poslocs. A depth error to the interior sensor now displays the following effect. As the sensor is moved deeper, the sound velocity of water will generally increase. This causes the elevation angles from each hydrophone to decrease when adjusted to the SBL posloc. This change in elevation angles, with respect to depth, will also be seen in the LBL poslocs for the same reason. Consistency will still be present even with this relationship between the elevation angles and depth.

c. Tilt Angles

The tilt errors, xtilt and ytilt, can be visualized as the orientation of the plane containing the X, Y, and C hydrophones. This plane will be defined by xtilt and ytilt angles from an origin centered at the C-hydrophone. In the presence of no other errors, the SBL poslocs will be at "truth" for the ACTUAL values of the tilts. It is in this manner that the correction will be made. By constructing two planes, one locating the SBL poslocs (erroneous) and the other locating the LBL poslocs ("truth"), the difference in plane tilts can be found. This difference is the correction to the OPERATIONAL xtilt and ytilt. For convenience, the regions closest to 0° and 90° (in the range coordinate system) are chosen, although any two non collinear combination regions will work.

d. X, Y, and Z Coordinate Errors

These three positional errors will be determined in an identical manner. Any of these coordinate errors will manifest itself in each region consistently, although not necessarily proportionally (see section 2b: Depth by Elevation Angle, for a summary of the unique depth effects). Errors from the comparison of LBL to SBL posloc positions, averaged over all regions will be used as a first guess correction to these coordinate errors. Since the effect of small angular errors produces large positional posloc errors, a fraction of this average error is used in order to guard against overshoot.

3. Integration of Error Corrections

The above individual error corrections schemes are listed in the order of importance to the error fixing (ERRFIX) algorithm. The approach chosen will correct for angular errors first, followed by positional errors. This choice is made due to the large displacement effect caused by the angular errors at the distances of interest.

The location corrections are preformed in an iterative fashion, correcting for one component error at a time. The new location of the array is evaluated after each correction, before proceeding on to the next correction. This allows for a controlled approach to the ACTUAL array location.

Figure 8 outlines the ERRFIX algorithm and figure 9 the detailed flow diagrams for the specified components. The tilt correction component of the first six error correction schemes, has a restriction placed on it. This restriction denies entry if no other previous corrections have been made. This minimizes the possibility of unnecessary cycling and invalid corrections to sensor location. The adjusted (SHIFT) tilt angles were found to move away from the ACTUAL tilt angles, in repeated back-to-back executions of the tilt correction component.

The tuning constant to the positional corrections is introduced prior to the individual rectangular coordinate corrections. Since the angular errors have not been totally adjusted as yet, the posloc positional errors contain

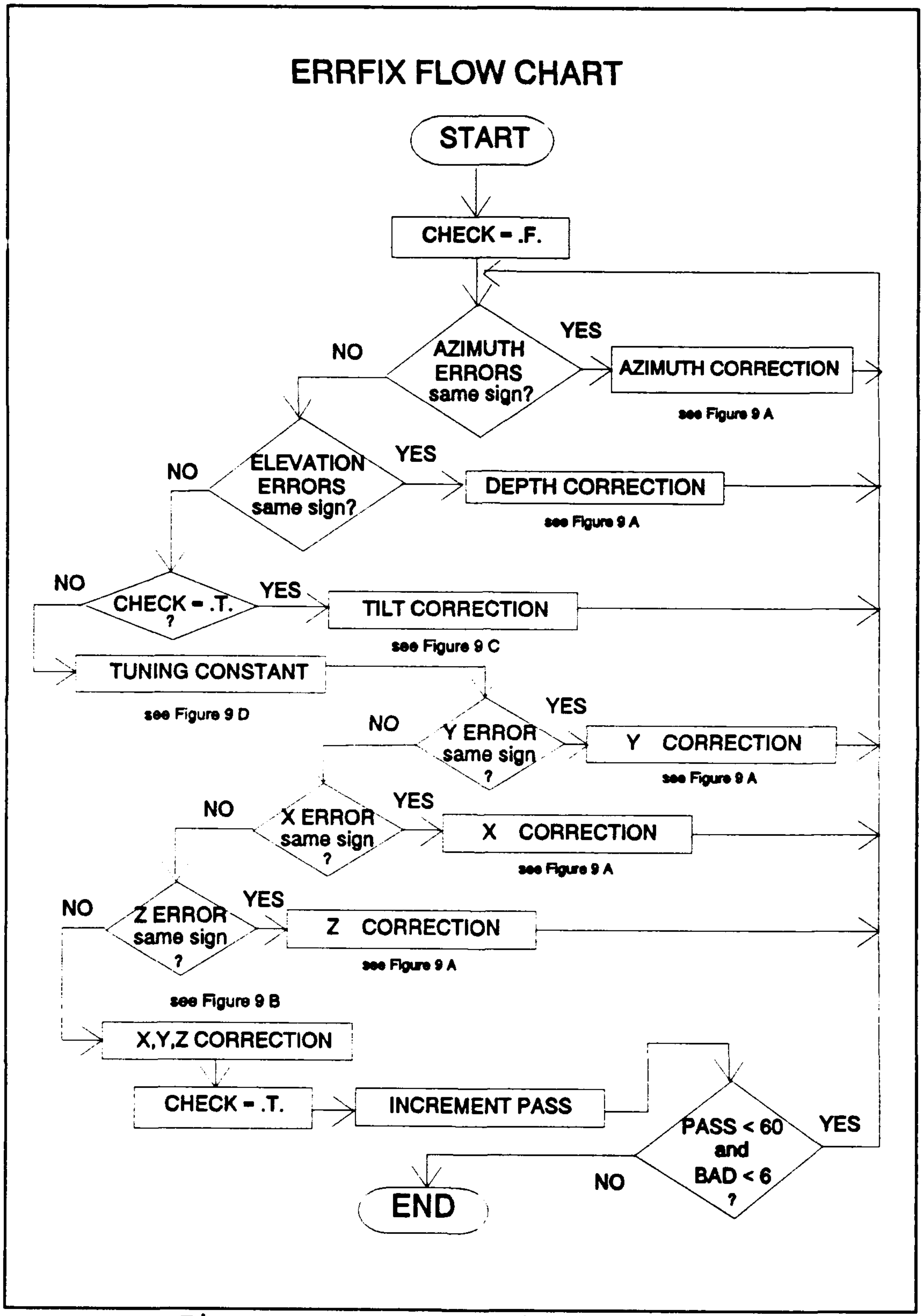

Figure 8. ERRFIX Logic Flow Chart.

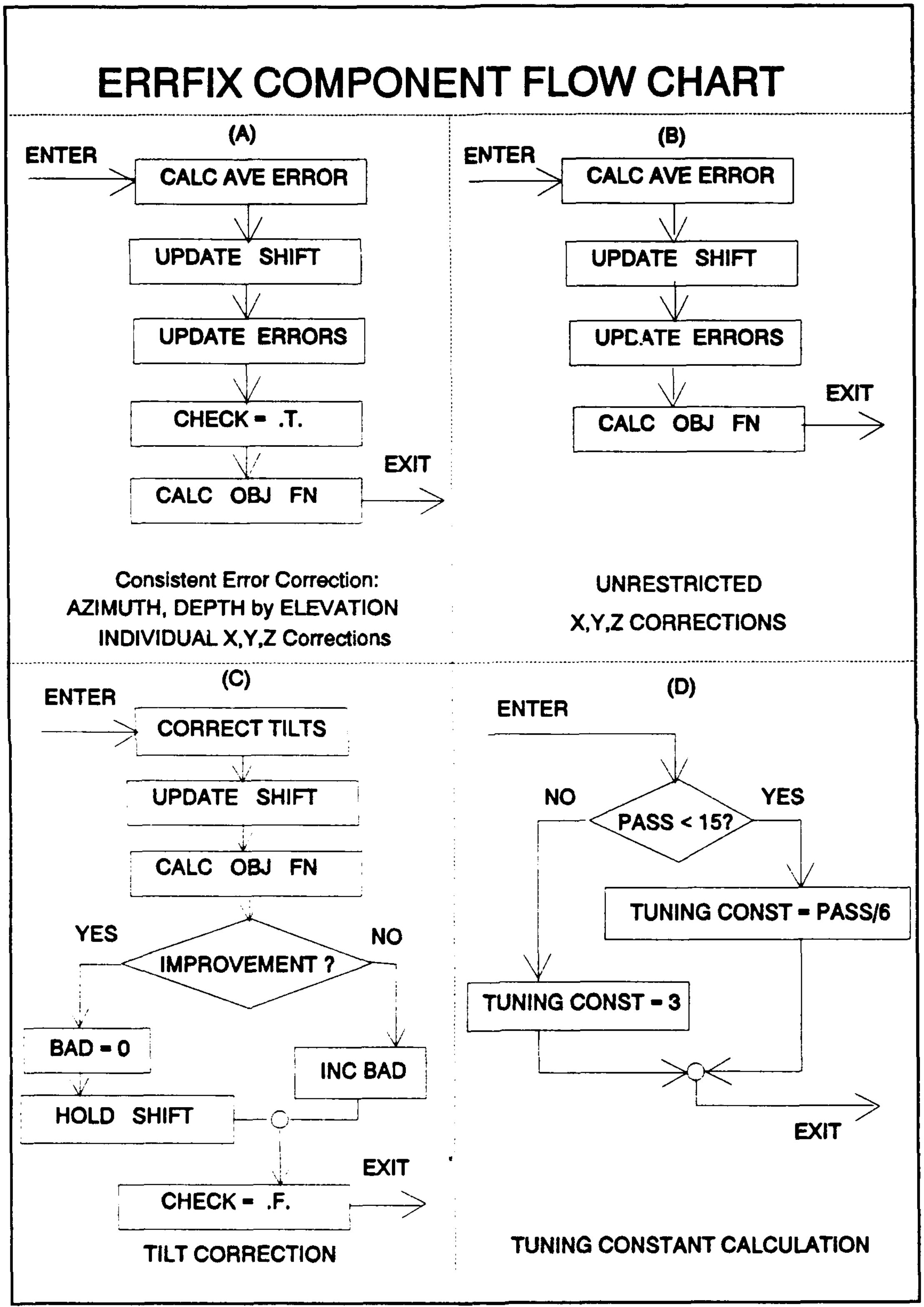

Figure 9. ERRFIX Individual Component Logic Flow Chart.

influences both from the angular and positional sensor locational errors. The adjustment of the positional errors using but a fraction of the average error causes the optimal, or ACTUAL, location to be approached more slowly. This is desirable in early iterations. As the number of passes through ERRFIX increases, the tuning constant is increased to speed convergence of the algorithm. As the ACTUAL location is approached, and the posloc errors reduced in magnitude, a fairly constant correction rate is then maintained. To limit this weighting scheme, so as not to overshoot the ACTUAL location, a ceiling of 3 for the tuning constant is imposed after 15 passes. The selection of these number was based on a minimal sensitivity analysis so efficiency is not expected.

Unrestricted position correction (x, y, and z) is allowed after all the consistent errors are removed. These correction schemes still use averaged posloc errors and the tuning constant in the computation of the SHIFT vector.

Exit criteria for ERRFIX is based on the number of passes made and the number of nonprofitable attempts to correct the sensor's location. The algorithm is exited after either 60 passes through ERRFIX, or 5 non improving attempts to correct location are made. Additional reasons (other than to prevent infinite cycling), for these exit criteria are presented in chapter IV.

F. LOCATION OPTIMIZATION (ANNEAL)

Since ERRFIX is a heuristic, an optimal solution is not guaranteed. A simulated annealing (ANNEAL) algorithm is then employed to further minimize the objective function. The basis for this algorithm is outlined in reference 8. The random number generator that is required came from the IMSL Library of Mathematical and Statistical functions [Ref. 9]. This library is developed for IBM PC computers and compiled in machine language and not reproducible here. Figure 10 displays ANNEAL's logic structure.

As ANNEAL is executed with the current solution (SHIFT) near the ACTUAL, a small step size to be applied to SHIFT is used. This small step size, while increasing run time, also helps to prevent overshoot of the optimal location. The numbers chosen to size the steps are based upon limited sensitivity analysis conducted on the objective function in the vicinity of the minimum. The validation check for the proposed step is to insure that the new angular values are within bounds (xtilt, ytilt; $-\pi/2$ to $\pi/2$, and zrot; $-\pi$ to π).

The 'cooling schedule' for the algorithm, which determines the probability of accepting a nonimproving step, is outlined in reference 8. A limited sensitivity analysis was performed to determine the parameters used in the calculation of the probabilities, and so efficiency may not be achieved.

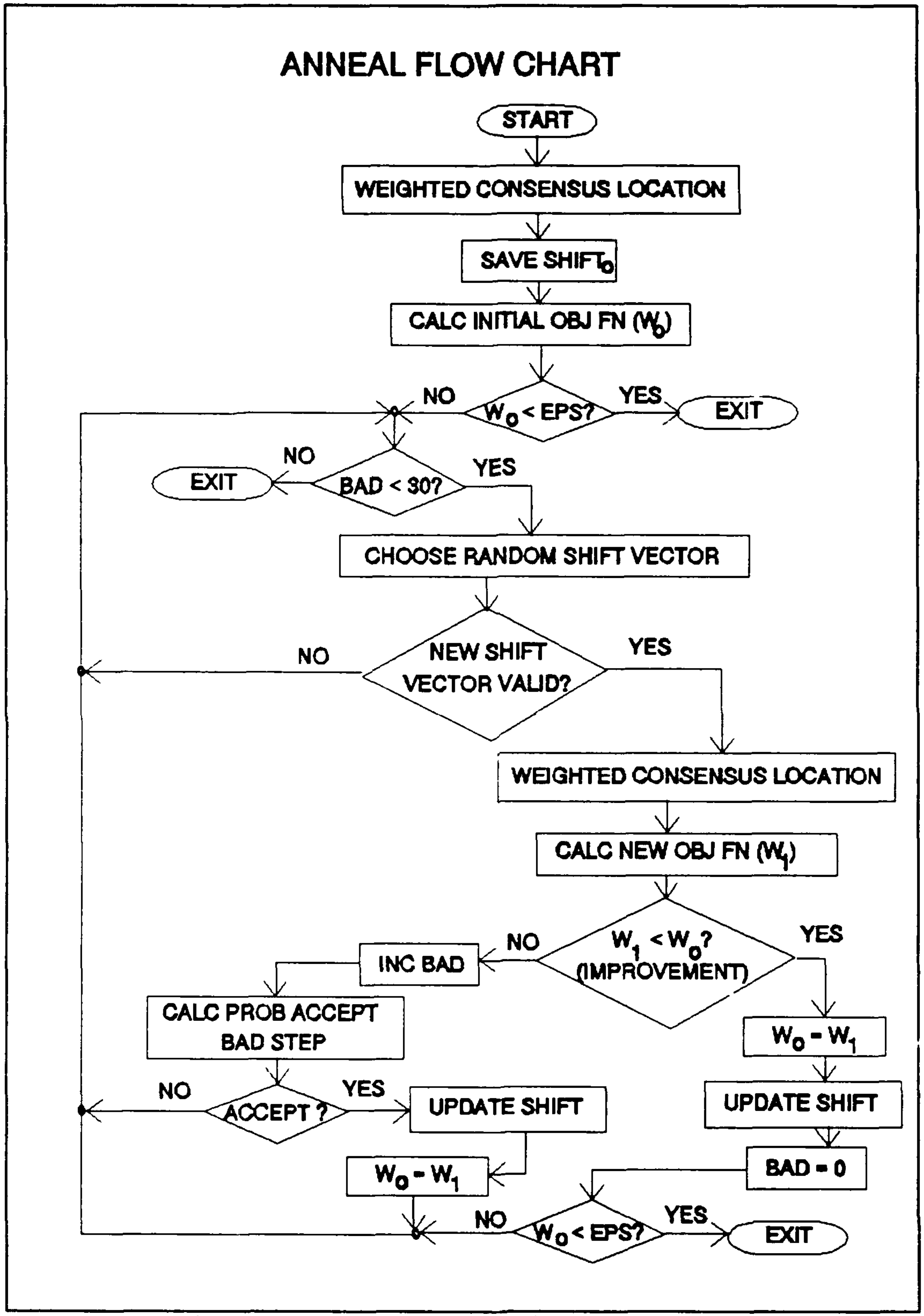

Figure 10. ANNEAL Logic Flow Chart.

Termination for the ANNEAL algorithm is two fold. Either 30 bad (nonimproving) steps accepted, or the objective function within epsilon (10^{-4}) of zero, will cause exiting.

IV. RANGE INTEGRATION (CONTROL)

Having the ability to correct for errors in an 'isolated' interior array (one with error free satellites), we turn to the problem of correcting several arrays. In order to accomplish this goal, the first assumption made, the interior array's satellites are error free, needs to be removed. To do this, an interior array will be chosen for correction, in which all of its satellite arrays are closer to their ACTUAL location than this interior one. The objective function presented on chapter III will be utilized to determine the array of interest. By choosing the array with the maximum objective function value (either range global, or with respect to its satellites), all of the satellite arrays are expected to require smaller locational adjustments.

A. CONTROL

The FORTRAN program, CONTROL, is set up to accomplish this range wide integration. Two problems arise when multiple arrays possess locational errors. First, in trying to achieve locational optimality for the array of interest, locational errors from the satellite arrays will prevent the objective function from reaching a zero value. Second, the sensing arrays that are selected for correction based on the objective function value, may not be interior sensors (fewer than six triple overlap regions, e.g. array 24 of Figure 3).

In the first problem, it was found that an iterative procedure was necessary to correct for multiple array location errors. Thus in ERRFIX, the restrictions on the number of passes and the number of 'bad' steps, was introduced (in addition to preventing infinite cycling). This allows for location improvement without over-improving towards possible erroneous LBL poslocs. After the array of interest exits from ERRFIX, the entire range is re-examined to find the new worst array, and the process repeated. Since correction is in a step wise fashion, entrance into ANNEAL is delayed until all arrays are 'close' to their ACTUAL locations. The logic flow chart for CONTROL is displayed Figure 11.

The other downside to 'all' the range arrays possessing locational errors, is that there will be no anchoring point for range location. The array locations will be optimized with respect to each other, but not with respect to any fixed outside point. The target tracks will then coincide in the overlap regions, but may not be the true track; only a relative track.

It was noted that arrays with induced locational errors produced objective function values significantly larger than the error free arrays. An attempt was made to take advantage of this trend. In singling out those arrays with the larger than average objective function values, the correction

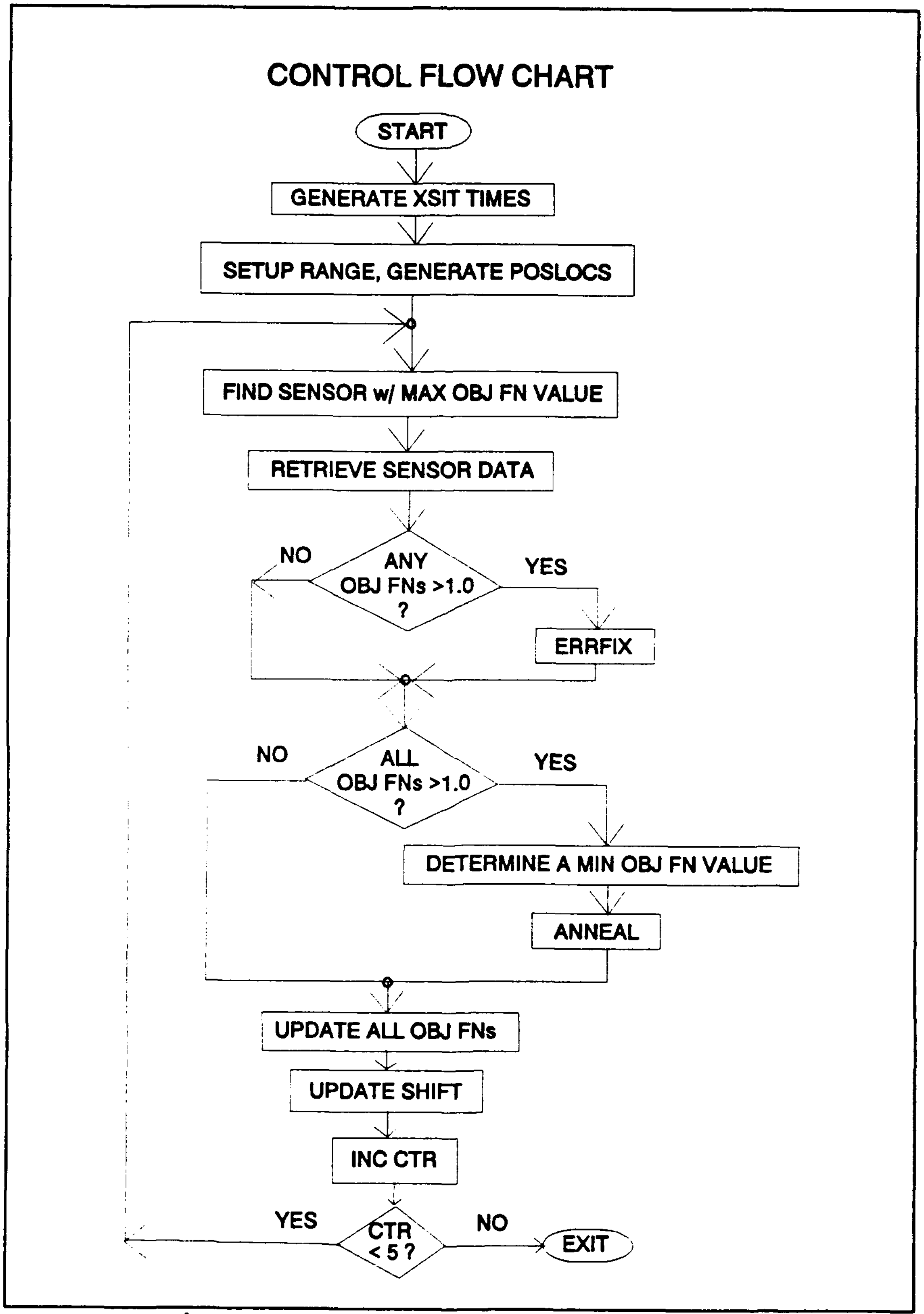

Figure 11. CONTROL Logic Flow Chart.

methodology could be selectively controlled. This decision rule has the advantage of reducing computer run time by eliminating the 'error free' arrays from the iteration scheme. The values of the control parameters were chosen arbitrarily.

The second problem addressed, involvement of arrays with fewer than six overlap regions, could not be studied fully due to time constraints. The problems that occur here can be seen by a study of array number 24. This array has its overlap regions all located on one side. This will cause the ERRFIX algorithm to induce invalid depth corrections caused by tilt errors. Any tilt errors that happen to displace all the SBL poslocs upward (negative elevation angle errors), will be seen as a depth error in the Depth by Elevation Angle component of ERRFIX. This effect will cause multiple optima to exist for these sensors without a full complement of regions.

V. RESULTS

A. ERROR FREE SATELLITES/ ISOSPEED DVP

In this ideal situation, sensor array 15 was chosen as the test case. A random location ERROR was then applied to the array location. The initial location chosen (ACTUAL), is the location currently defined by the Nanoose range. Application of the ERROR vector to this ACTUAL location generated our OPERATIONAL location. This data is presented below.

Table I. SENSOR ARRAY 15 LOCATIONAL DATA.

COMPONENT	ACTUAL LOCATION	INDUCED ERROR	RESULTANT OPERATIONAL LOCATION
X (ft)	53249.43	11.00	53260.43
Y (ft)	-6354.60	28.00	-6326.60
Z (ft)	1316.66	-2.00	1314.66
xtilt (radians)	0.003345	0.0250	0.028345
ytilt (radians)	0.004509	0.0100	0.014509
zrot (radians)	0.581544	-0.0100	0.571544

Shown below (Figure 12), is a two dimensional depiction of sensor array 15 in both the ACTUAL and OPERATIONAL locations.

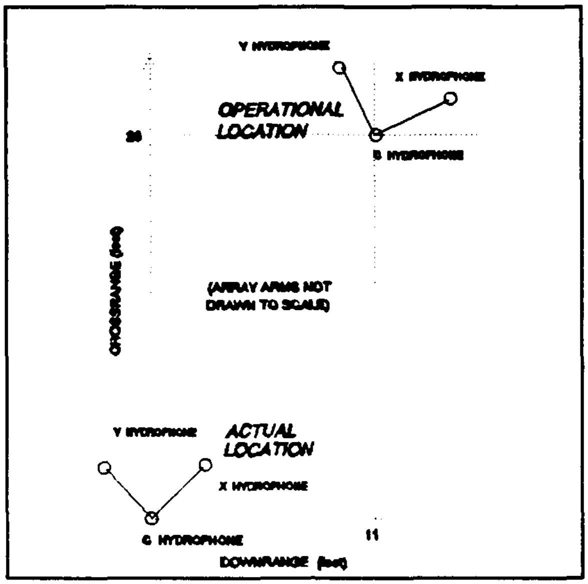

Figure 12. ACTUAL and OPERATIONAL Location of Sensor Array 15.

In evaluating this array, the triple overlap regions associated with it will be defined in according to the following numbering scheme (refer to Figure 3):

SERVING ARRAYS	REGION
15,14,24	13
15, 5,14	14
15, 5, 6	15
15, 6,16	21
15,16,25	22
15,24,25	23

With the above numbering system, sensor array 15 (OPERATIONAL), and its six satellites produce the following posloc data by region:

REGION 13

	ARRAY 15	ARRAY 14	ARRAY 24	LONGBASE	TRUTH
X posloc	49502.98	49493.30	49493.30	49506.25	49493.30
Y posloc	-8599.35	-8593.79	-8593.79	-8586.09	-8593.79
Z posloc	485.07	400.00	400.00	436.63	400.00
PHS	-2.60112	-0.50009	1.58474		
PHL	-2.60333	-0.49714	1.58175		
THS	0.19055	0.21122	0.20692		
THL	0.20163	0.20287	0.19853		
HS	4376.31	4383.21	4362.25		
HL	4366.69	4390.90	4369.78		

REGION 14

	ARRAY 15	ARRAY 5	ARRAY 14	LONGBASE	TRUTH
X posloc	49371.89	49404.92	49404.92	49402.52	49402.92
Y posloc	-4363.58	-4356.84	-4356.84	-4356.84	-4356.84
Z posloc	510.17	400.00	400.00	393.66	400.00
PHS	2.66930	-1.59591	0.51668		
PHL	2.66441	-1.59648	0.51724		
THS	0.18565	0.21220	0.21411		
THL	0.21244	0.21367	0.21557		
HS	4361.03	4319.56	4322.25		
HL	4337.52	4318.19	4320.87		

REGION 15

	ARRAY 15	ARRAY 5	ARRAY 6	LONGBASE	TRUTH
X posloc	53178.27	53235.75	53235.75	53235.67	53235.75
Y posloc	-2053.41	-2085.69	-2085.69	-2066.51	-2085.69
Z posloc	421.31	400.00	400.00	358.59	400.00
PHS	1.58878	-0.50279	-2.63487		
PHL	1.57543	-0.49884	-2.63884		
THS	0.20833	0.21566	0.21393		
THL	0.22296	0.22542	0.22368		
HS	4294.82	4248.10	4249.12		
HL	4281.08	4238.82	4239.91		

REGION 21

	ARRAY 15	ARRAY 6	ARRAY 16	LONGBASE	TRUTH
X posloc	57018.62	57054.98	57054.98	57067.90	57054.98
Y posloc	-4259.21	-4330.68	-4330.68	-4322.70	-4330.68
Z posloc	308.48	400.00	400.00	364.99	400.00
PHS	0.50664	-1.54664	2.65435		
PHL	0.48823	-1.54360	2.65132		
THS	0.23287	0.21107	0.21130		
THL	0.21976	0.21920	0.21939		
HS	4303.74	4308.39	4329.57		
HL	4316.75	4300.74	4321.91		

REGION 22

	ARRAY 15	ARRAY 16	ARRAY 25	LONGBASE	TRUTH
X posloc	57060.74	57054.15	57054.15	57051.49	57054.15
Y posloc	-8516.21	-8587.96	-8587.96	-8586.52	-8587.96
Z posloc	285.63	400.00	400.00	407.54	400.00
PHS	-0.51801	-2.61390	1.55241		
PHL	-0.53291	-2.61448	1.55302		
THS	0.23391	0.20669	0.18370		
THL	0.20617	0.20498	0.18199		
HS	4379.83	4429.02	4416.12		
HL	4407.10	4430.60	4417.52		

REGION 23

	ARRAY 15	ARRAY 24	ARRAY 25	LONGBASE	TRUTH
X posloc	53322.58	53293.11	53293.11	53293.83	53293.11
Y posloc	-10673.91	-10707.34	-10707.34	-10689.88	-10707.34
Z posloc	373.46	400.00	400.00	447.16	400.00
PHS	-1.55529	0.54138	2.58376		
PHL	-1.56197	0.54471	2.58026		
THS	0.21740	0.20689	0.18696		
THL	0.20040	0.19609	0.17609		
HS	4326.89	4362.89	4337.38		
HL	4342.51	4372.52	4346.04		

Summarizing the above data into the error table gives the following:

TABLE II. ERROR TABLE FOR SENSOR ARRAY 15 (ERROR FREE SATELLITES/ISOSPEED DVP)

	REGION					
Attribute	13	14	15	21	22	23
THER	0.01108	0.02679	0.01463	-0.01311	-0.02775	-0.01700
PHER	-0.00222	-0.00489	-0.01335	-0.01840	-0.01940	-0.00668
HER	-9.622	-23.501	-13.739	13.010	27.266	15.624
XER	3.278	30.627	57.394	49.280	-9.248	-28.753
YER	13.258	8.178	-13.091	-63.494	-70.306	-15.975
ZER	-48.436	-116.508	-62.719	56.502	121.910	73.699

This test case shows the depth exaggeration produced by LONGBASE. The poslocs generated do not approximate those normally encountered on the range. This is not expected since only one array is mispositioned while its satellites are error free.

Submitting this data to ERRFIX the following solution is returned after 13 passes. (The algorithm was terminated on the 5 bad steps criteria, so only 8 passes were needed to produce the solution.)

TABLE III. SENSOR ARRAY 15 ERROR CORRECTION DATA (ERRFIX). ERROR FREE SATELLITES - ISOSPEED DVP.

COMPONENT	INDUCED ERROR	CORRECTED SHIFT
X (ft)	11.000	-11.000
Y (ft)	28.000	-28.000
Z (ft)	-2.000	2.000
xtilt (radians)	0.02500	-0.02500
ytilt (radians)	0.01000	-0.01000
zrot (radians)	-0.01000	0.01000

The solution presented above for array 15 is accurate to the thousandth of a foot and less than a hundred thousandth of a radian. The above solution was achieved without utilizing the ANNEAL algorithm.

Several other test cases for array 15 were run to verify the algorithm following the same format as above. The results of the induced ERROR and the corrected SHIFT are presented on the next page (Table IV).

TABLE IV. SENSOR ARRAY 15 ERROR CORRECTION TEST CASES (ERRFIX). ERROR FREE SATELLITES — ISOSPEED DVP.

	ERROR	SHIFT	ERROR	SHIFT	ERROR	SHIFT	ERROR	SHIFT
x (ft)	-11.000	10.999	10.000	-10.000	-15.000	15.000	5.000	-5.000
y (ft)	-28.000	27.999	-12.000	11.999	11.000	-11.000	-2.000	2.000
z (ft)	2.000	-2.001	7.000	-7.000	-10.000	10.000	10.000	-10.000
xtilt (radians)	-0.02500	0.02500	-0.00600	0.00600	0.00600	-0.00600	-0.00100	0.00100
ytilt (radians)	-0.01000	0.01000	-0.00400	0.00400	0.00400	-0.00400	0.00400	-0.00400
zrot (radians)	0.01000	-0.01000	0.00700	-0.00700	-0.00800	0.00800	0.00300	-0.00300

B. INTERIOR ARRAY CORRECTION (SATELLITES WITH ERRORS/ ISOSPEED DVP)

The ERRFIX and ANNEAL algorithms are now checked for performance on an interior array whose satellites contain locational errors. Array 15 is again chosen as the array of interest. The ERRORs induced on the arrays are such that array 15 will have the largest objective function with respect to its satellites. Table V on the next page, displays the ERRORs induced for all the arrays. These ERRORs were chosen so as to obtain initial posloc data similar to that normally seen on the range. Appendix H displays the regional posloc data.

Submission of this data to ERRFIX produced the following corrections to array 15 (Table VI) after one pass. ERRFIX was exited after 11 passes, the last five non improving. With the exception of the ytilt component, the locational errors are reduced by 65 to 97%. The ytilt component was over corrected and increased its error magnitude by 142%. Table VI also shows the cumulative corrections produced by ANNEAL, using the ERRFIX corrections as input. The y and ytilt component errors are improved in ANNEAL while the other component errors are degraded. This supports the previous assessment of iteratively correcting the arrays iteratively through ERRFIX before attempting ANNEAL.

TABLE V. INDUCED ERRORS FOR INTERIOR ARRAY 15 AND ITS SATELLITE ARRAYS. (ISOSPEED DVP).

	ARRAY 15	ARRAY 14	ARRAY 24	ARRAY 5	ARRAY 6	ARRAY 16	ARRAY 25
X (ft)	11.000	4.000	-0.500	7.900	-5.000	5.000	1.500
Y (ft)	28.000	-8.000	-1.000	-6.200	7.000	-2.000	-1.200
Z (ft)	-2.000	5.000	0.700	-1.800	4.000	4.000	1.000
xtilt (radians)	0.00400	-0.00600	-0.00600	0.00300	0.00200	0.00300	0.00300
ytilt (radians)	0.00500	-0.00300	0.00200	-0.00400	0.00100	0.00200	-0.00300
zrot (radians)	-0.01000	0.00200	-0.00300	0.00100	-0.00300	-0.00100	0.00100

TABLE VI. ACCUMULATED CORRECTIONS FOR ARRAY 15. ERROR INDUCED SATELLITES - ISOSPEED DVP.

COMPONENT	INDUCED ERROR	ERRFIX CORRECTION	ANNEAL CORRECTION
X (ft)	11.000	-11.756	-15.789
Y (ft)	28.000	-35.444	-29.434
Z (ft)	-2.000	2.706	4.577
xtilt (radians)	0.00400	-0.00444	-0.00727
ytitlt (radians)	0.00500	-0.01209	-0.00714
zrot (radians)	-0.01000	0.00973	0.00942

C. CORRECTION OF MULTIPLE INTERIOR ARRAYS (ISOSPEED DVP)

After verifying the ERRFIX and ANNEAL algorithms work properly, correction for multiple interior arrays are now preformed. ERRORs are induced in the six interior arrays (see Figure 3) of the range, 4, 5, 6, 14, 15, and 16. Table VII displays the induced ERRORs. The exterior arrays are all properly located in this example.

It was necessary to place a 'no repeat' restriction on the arrays for entry into ERRFIX. This restriction prevented the CONTROL algorithm from cycling on one single array. Iterative passes through ERRFIX was terminated when all the objective

TABLE VII. INDUCED ERRORS FOR THE RANGE INTERIOR ARRAYS. (ISOSPEED DVP).

	ARRAY 4	ARRAY 5	ARRAY 6	ARRAY 14	ARRAY 15	ARRAY 16
X (ft)	-0.500	7.900	-5.000	4.000	11.000	5.000
Y (ft)	-1.000	-6.200	7.000	-8.000	28.000	-2.000
Z (ft)	0.700	-1.800	4.000	5.000	-2.000	4.000
xtilt (radians)	-0.00600	0.00300	0.00500	-0.00900	0.02500	0.00400
ytilt (radians)	0.00300	-0.00400	0.00200	-0.00400	0.01000	0.00200
zrot (radians)	-0.00200	0.00100	-0.00300	0.00300	-0.01000	0.00100

functions were less than one. The choice of one was arbitrarily chosen to speed convergence (entry to ANNEAL). Due to the consistency of errors approach in ERRFIX, as the ACTUAL location is approached, the magnitude of the correction is decreased. Presented below (Tables VIII - XIII) are the corrections (SHIFTS) produced from ERRFIX at the termination criteria mentioned above.

TABLE VIII. ACCUMULATED CORRECTIONS FOR ARRAY 4. ERROR INDUCED INTERIOR ARRAYS - ISOSPEED DVP.

COMPONENT	INDUCED ERROR	ERRFIX CORRECTION	ANNEAL CORRECTION
X (ft)	-0.500	0.420	0.506
Y (ft)	-1.000	0.664	0.980
Z (ft)	0.700	-0.589	0.668
xtilt (radians)	-0.00600	0.00594	0.00599
ytitlt (radians)	0.00300	-0.00288	-0.00298
zrot (radians)	-0.00200	0.00206	0.00200

TABLE IX. ACCUMULATED CORRECTIONS FOR ARRAY 5. ERROR INDUCED INTERIOR ARRAYS - ISOSPEED DVP.

COMPONENT	INDUCED ERROR	ERRFIX CORRECTION	ANNEAL CORRECTION
X (ft)	7.900	-7.728	-7.978
Y (ft)	-6.200	5.563	6.159
Z (ft)	-1.800	2.846	1.939
xtilt (radians)	0.00700	-0.00677	-0.00702
ytitlt (radians)	-0.00300	0.00294	0.00298
zrot (radians)	0.00100	-0.00099	-0.00100

TABLE X. ACCUMULATED CORRECTIONS FOR ARRAY 6. ERROR INDUCED INTERIOR ARRAYS - ISOSPEED DVP.

COMPONENT	INDUCED ERROR	ERRFIX CORRECTION	ANNEAL CORRECTION
X (ft)	-5.000	5.373	4.975
Y (ft)	7.000	-7.936	-7.009
Z (ft)	4.000	-2.754	3.808
xtilt (radians)	0.00500	-0.00499	-0.00498
ytitlt (radians)	0.00200	-0.00243	-0.00200
zrot (radians)	-0.00300	0.00297	0.00300

TABLE XI. ACCUMULATED CORRECTIONS FOR ARRAY 14. ERROR INDUCED INTERIOR ARRAYS - ISOSPEED DVP.

COMPONENT	INDUCED ERROR	ERRFIX CORRECTION	ANNEAL CORRECTION
X (ft)	4.000	-3.941	-4.027
Y (ft)	-8.000	8.009	7.994
Z (ft)	5.000	-4.917	4.994
xtilt (radians)	-0.00900	0.00915	0.00901
ytitlt (radians)	-0.00400	0.00392	0.00401
zrot (radians)	0.00300	-0.00298	-0.00300

TABLE XII. ACCUMULATED CORRECTIONS FOR ARRAY 15. ERROR INDUCED INTERIOR ARRAYS - ISOSPEED DVP.

COMPONENT	INDUCED ERROR	ERRFIX CORRECTION	ANNEAL CORRECTION
X (ft)	11.000	-10.650	-11.029
Y (ft)	28.000	-28.858	-28.019
Z (ft)	-2.000	1.149	2.064
xtilt (radians)	0.02500	-0.02451	-0.02499
ytitlt (radians)	0.01000	-0.01020	-0.01002
zrot (radians)	-0.01000	0.01004	0.01000

TABLE XIII. ACCUMULATED CORRECTIONS FOR ARRAY 16. ERROR INDUCED INTERIOR ARRAYS - ISOSPEED DVP.

COMPONENT	INDUCED ERROR	ERRFIX CORRECTION	ANNEAL CORRECTION
X (ft)	5.000	-5.203	-5.004
Y (ft)	-2.000	1.031	2.001
Z (ft)	4.000	-5.132	-3.974
xtilt (radians)	0.00400	-0.00336	-0.00400
ytitlt (radians)	0.00200	-0.00192	-0.00200
zrot (radians)	0.00100	-0.00102	-0.00100

ERRFIX corrected well for all locational components except depth. The corrections provided by ERRFIX left errors of at most 58% remaining (depth correction for array 5). The only other locational component approaching this magnitude is the y component for array 16 (48% error remaining). Internal passes through ERRFIX provided for improvements in all cases before termination (on 5 non improving passes).

Once all the interior array objective functions fell below one, control was passed to ANNEAL. The restriction of no 'back to back' corrections for an array was also in place here. Convergence within ANNEAL was generally quick, with termination occurring on 30 consecutive nonimproving steps accepted in the search. In order to maximize the potential for optimization, an internal control structure was added to ANNEAL to increase the step size and then decrease the bad step acceptance probability. As soon as a lower objective function value is found (over the previous minimum), the parameters are returned to their initial values. ANNEAL termination occurs after 1500 passes with the larger step size and lower probability, if a lower objective function value has not been found. This exit condition was only invoked three times in the above test case. The results of ANNEAL performance (cumulative corrections) are displayed along with the ERRFIX corrections in Tables VIII to XIII.

CONTROL was terminated when any array's objective function value increased after a return from ANNEAL. This occurred with array 15, after a total of 33 array entrances into ANNEAL (this was array 15's seventh entry to ANNEAL).

VI. CONCLUSIONS AND RECOMMENDATIONS

A. RESULTS

The goal of this thesis was to investigate interior sensor array locational errors, and develop a set of decision rules to correct any errors that may exist. The primary tools used here include, long baseline methodology and a consistency of errors approach.

It was found that although LONGBASE was written under the caveat of no positional array errors existing, it could still be used to aid in correction of an array's location. In fact it worked better than using an averaged consensus posloc (of the three SBL poslocs) to compare the SBL poslocs against.

The current research did not assume that the exterior arrays (those with fewer than six triple overlap regions) were locationally in error. Investigation of their inclusion into the error correction methodology was not conducted, so their effect on the algorithms is undetermined. The test cases r~~a~~un in this study also used an isospeed DVP. Appendix A addresses the effects of a DVP containing a gradient. The resultant gradient induced errors resulting from the use of Professor Read's recommended ray tracing initialization method, are shown to be small in magnitude. Time constraints prevented a study of their effect on the error correction scheme.

The final algorithms, ERRFIX and ANNEAL, have demonstrated excellent robustness in the correction of locational errors. An annealing algorithm, by itself, may be able to successfully minimize array locational errors. Since the array's location is six dimensional, it was deemed unreasonable to rely solely on this technique in order to keep computer run time down. The error correction algorithms were written so as to be able to accept raw tracking data from the range. The data used in the study here was internally generated to eliminate any systematic error sources commonly found in an underwater tracking situation. Times provided by the range are generally suspect in themselves, thus preventing 'error free' posloc determination.

B. RECOMMENDATIONS FOR FURTHER RESEARCH

In expanding this technique to the entire tracking range, three problems need to be analyzed.

First, there is the treatment of exterior arrays. The use of these arrays in the current methodology may have insufficient consistency of errors to provide locating information. The use of double overlap regions [Refs. 4,5, and 6], may be necessary to supply enough data to accurately correct these exterior arrays.

Second, the optimization of the internal control parameters in the ERRFIX, ANNEAL and CONTROL algorithms is necessary before attempting entire range correction. This will also reduce computer run time allowing for more frequent calibration updates. The run times in solving for a single array with error free satellites was on the order of 5 minutes. The case of solving for all the interior arrays required about one and half hours.

The third area of work, is the effect of gradient water environments. Although the errors seen resulting from the gradients are 'small', the effect on an iterative correction scheme may be multiplictative thus inhibiting the correction of the true locational errors.

APPENDIX A: GRADIENT DVPs AND RAY TRACING INITIALIZATION

An isospeed DVP was used during the formulation of this correction method. This had the advantage of eliminating extraneous errors from the analysis.

About third of posloc errors were estimated to be due to the ray tracing initialization in a gradient DVP environment. The study performed in reference 1, showed that the current ray tracing initialization method used by NUWES produced errors on the order of feet in depth and horizontal range, and third decimal place (radians) in elevation and azimuth angles. The method recommended by Professor Read reduced these errors by a factor of 10 for depth, horizontal range, and elevation angle, and a factor of 100 for azimuth angle.

Figure 4 of reference 1, showed a sinusoidal nature to each of the above error components. The skewed nature of the curves produced made direct analysis difficult. Standardizing the array axis (setting xtilt = ytilt = zrot = 0) produced the following curves (Figure 13). This figure mimics Figure 4 of reference 1, array 1 with the DVP of 3/22/89 used for sound sources at 3000, 4000 and 5000 feet.

With a standardized local coordinate system the sinusoidal nature of the errors becomes visible. Analysis of the angular errors (azimuth and elevation angles) shows the errors to be 90° out of phase. The zero error points for azimuthal error

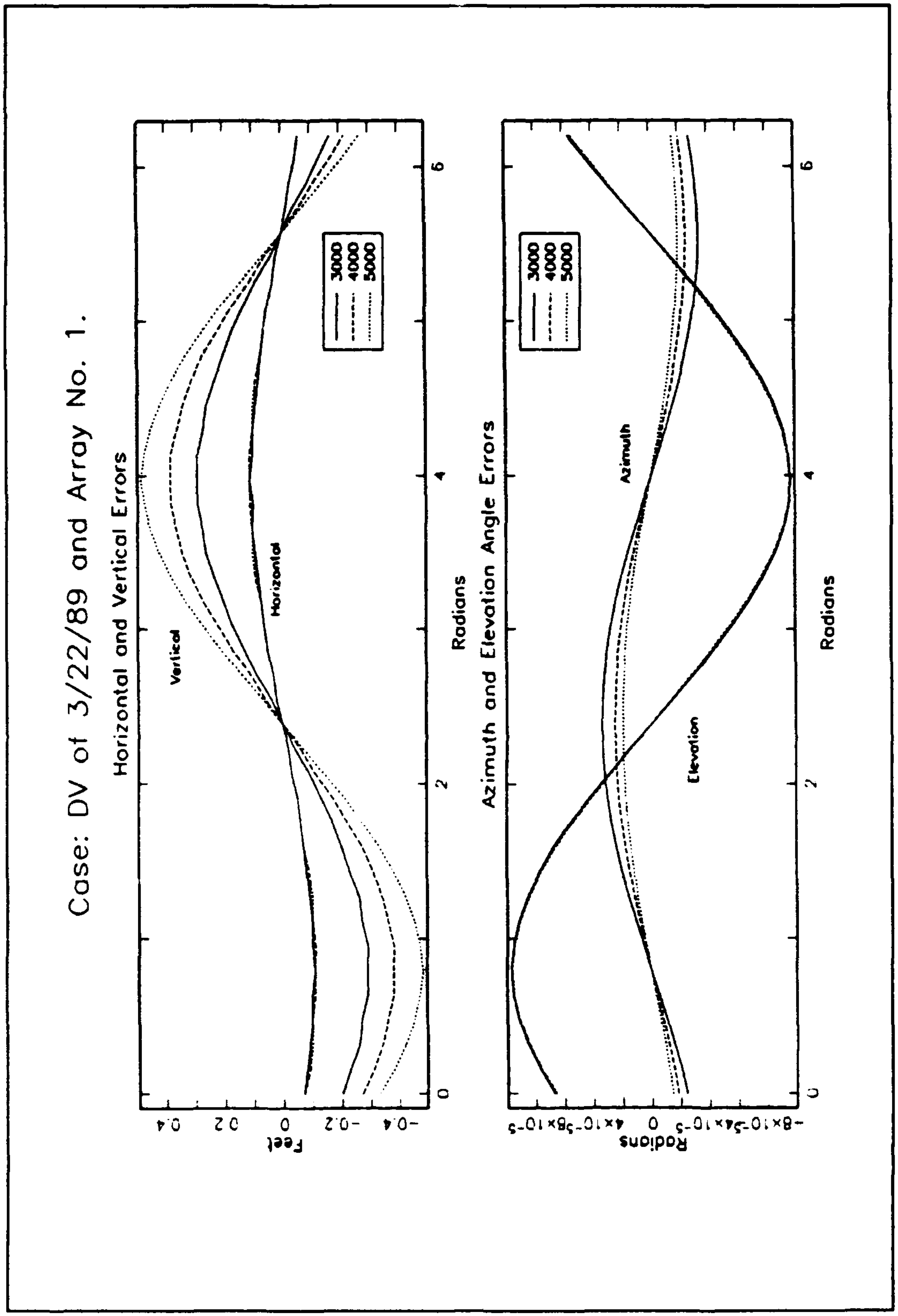

Figure 13. Gradient Induced Sinusoidal Errors.

occurs at 45° and 235° rotation (CCW from east, the x axis). This corresponds to the point where the sound source is equidistant from both the x and y hydrophones. The maximum azimuthal error occurs at 135° and 325°, 90° out from the zero errors. These points are where the sound source has maximum distance to one of the hydrophones (x or y), and minimum distance to the other. The elevation angles display the opposite phenomenon, maximum errors where the sound source is equidistant form the x and y hydrophones, and zero error at the maximum/minimum distance to one or the other.

This effect on the errors is traced back to the geometric construction of the posloc point to initialize Professor Read's ray tracing method. A constant speed of sound in water is used to construct this posloc point. This speed of sound used (the speed at the level of the x, y, and c hydrophones) is larger than the speed found along the ray path. (Speed of sound in water generally decreases as you get shallower.) This larger speed of sound used causes the distance traveled back along the ray path to be longer than it is in reality. Thus the sinusoidal nature of the errors occur with respect to the array axis, or the sides of the pyramid constructed to locate the posloc. The discrepancies observed in figure 13 are the result of time differences (time from constructed ray paths versus real transit times) in the ninth and tenth decimal place.

Before proceeding in an attempt to correct the effect, the posloc errors produced are examined. Using array 15, with ERROR free satellites and an isogradient DVP (+ 0.08 ft/sec), the following correction data is produced (Table XIV).

TABLE XIV. CORRECTIONS FOR ARRAY 15 ERROR INDUCED SATELLITES - ISOGRADIENT DVP.

COMPONENT	INDUCED ERROR	ERRFIX CORRECTION	ANNEAL CORRECTION
X (ft)	11.000	-11.001	-11.001
Y (ft)	28.000	-28.003	-28.003
Z (ft)	-2.000	1.994	1.994
xtilt (radians)	0.02500	-0.02505	-0.02505
ytitlt (radians)	0.01000	-0.01005	-0.01005
zrot (radians)	-0.01000	0.01000	0.01000

The following regional posloc data is provided to give the reader an idea as to the effect this gradient has on the poslocs. The magnitude of the posloc errors are proportional to the gradient of the DVP.

REGIONAL POSLOC DATA BEFORE ERRFIX

REGION 13

	ARRAY 15 SBL	ARRAY 14 SBL	ARRAY 24 SBL	LONGBASE	GOAL
X	49502.76	49493.37	49493.35	49506.26	49493.30
Y	-8600.14	-8593.79	-8593.85	-8586.09	-8593.79
Z	484.96	400.27	399.72	435.37	400.00
THS	0.18338	0.20397	0.19983		
THL	0.19475	0.19595	0.19164		
PHS	-2.60099	-0.50009	1.58473		
PHL	-2.60333	-0.49714	1.58175		
HS	4376.91	4383.27	4362.19		
HL	4366.68	4390.91	4369.79		

REGION 14

	ARRAY 15 SBL	ARRAY 5 SBL	ARRAY 14 SBL	LONGBASE	GOAL
X	49371.31	49404.88	49404.99	49402.53	49404.92
Y	-4363.06	-4356.78	-4356.83	-4355.41	-4356.84
Z	509.94	399.77	400.26	393.90	400.00
THS	0.17852	0.20517	0.20696		
THL	0.20527	0.20654	0.20844		
PHS	2.66925	-1.59592	0.51667		
PHL	2.66441	-1.59647	0.51724		
HS	4361.78	4319.51	4322.31		
HL	4337.51	4318.20	4320.87		

REGION 15

	ARRAY 15 SBL	ARRAY 5 SBL	ARRAY 6 SBL	LONGBASE	GOAL
X	53179.04	53235.71	53235.73	53235.67	53235.75
Y	-2053.32	-2085.64	-2085.75	-2066.52	-2085.69
Z	421.03	399.72	400.19	359.91	400.00
THS	0.20135	0.20876	0.20691		
THL	0.21564	0.21817	0.21643		
PHS	1.58860	-0.50279	-2.63486		
PHL	1.57543	-0.49884	-2.63884		
HS	4294.90	4248.04	4249.16		
HL	4281.07	4238.83	4239.92		

REGION 21

	ARRAY 15 SBL	ARRAY 6 SBL	ARRAY 16 SBL	LONGBASE	GOAL
X	57018.23	57054.97	57054.96	57067.89	57054.98
Y	-4260.02	-4330.75	-4330.62	-4322.71	-4330.68
Z	308.33	400.30	400.20	366.14	400.00
THS	0.22589	0.20394	0.20415		
THL	0.21243	0.21190	0.21205		
PHS	0.50651	-1.54665	2.65434		
PHL	0.48823	-1.54360	2.65132		
HS	4303.00	4308.46	4329.61		
HL	4316.74	4300.75	4321.92		

REGION 22

	ARRAY 15 SBL	ARRAY 16 SBL	ARRAY 25 SBL	LONGBASE	GOAL
X	57060.09	57054.14	57054.14	57051.48	57054.15
Y	-8515.56	-8587.89	-8588.02	-8586.51	-8587.96
Z	285.62	399.90	399.69	407.29	400.00
THS	0.22678	0.19945	0.17652		
THL	0.19899	0.19777	0.17480		
PHS	-0.51795	-2.61391	1.55241		
PHL	-0.53290	-2.61448	1.55302		
HS	4378.94	4429.00	4416.06		
HL	4407.09	4430.60	4417.53		

REGION 23

	ARRAY 15 SBL	ARRAY 24 SBL	ARRAY 25 SBL	LONGBASE	GOAL
X	53323.36	53293.16	53293.10	53293.81	53293.11
Y	-10673.68	-10707.40	-10707.39	-10689.86	-10707.34
Z	373.44	400.05	399.89	445.45	400.00
THS	0.21032	0.19972	0.17987		
THL	0.19366	0.18930	0.16935		
PHS	-1.55511	0.54136	2.58377		
PHL	-1.56197	0.54472	2.58026		
HS	4326.67	4362.90	4337.36		
HL	4342.48	4372.52	4346.07		

REGIONAL POSLOC DATA AFTER ERRFIX

REGION 13

	ARRAY 15 SBL	ARRAY 14 SBL	ARRAY 24 SBL	LONGBASE	GOAL
X	49493.30	49493.37	49493.35	49493.30	49493.30
Y	-8593.79	-8593.79	-8593.85	-8593.79	-8593.79
Z	399.99	400.27	399.72	400.00	400.00
THS	0.20342	0.20397	0.19983		
THL	0.20342	0.20403	0.19976		
PHS	-2.60761	-0.50009	1.58473		
PHL	-2.60761	-0.50009	1.58474		
HS	4358.54	4383.27	4362.19		
HL	4358.55	4383.20	4362.25		

REGION 14

	ARRAY 15 SBL	ARRAY 5 SBL	ARRAY 14 SBL	LONGBASE	GOAL
X	49404.92	49404.88	49404.99	49404.92	49404.92
Y	-4356.84	-4356.78	-4356.83	-4356.84	-4356.84
Z	400.00	399.77	400.26	400.00	400.00
THS	0.20440	0.20517	0.20696		
THL	0.20440	0.20512	0.20702		
PHS	2.65763	-1.59592	0.51667		
PHL	2.65763	-1.59591	0.51668		
HS	4338.39	4319.51	4322.31		
HL	4338.39	4319.56	4322.25		

REGION 15

	ARRAY 15 SBL	ARRAY 5 SBL	ARRAY 6 SBL	LONGBASE	GOAL
X	53235.75	53235.71	53235.73	53235.75	53235.75
Y	-2085.69	-2085.64	-2085.75	-2085.69	-2085.69
Z	400.01	399.72	400.19	400.00	400.00
THS	0.20680	0.20876	0.20691		
THL	0.20680	0.20870	0.20696		
PHS	1.57297	-0.50279	-2.63486		
PHL	1.57297	-0.50279	-2.63487		
HS	4289.76	4248.04	4249.16		
HL	4289.76	4248.10	4249.12		

REGION 21

	ARRAY 15 SBL	ARRAY 6 SBL	ARRAY 16 SBL	LONGBASE	GOAL
X	57054.98	57054.97	57054.96	57054.98	57054.98
Y	-4330.68	-4330.75	-4330.62	-4330.68	-4330.68
Z	400.00	400.30	400.20	400.00	400.00
THS	0.20511	0.20394	0.20415		
THL	0.20511	0.20401	0.20420		
PHS	0.49257	-1.54665	2.65434		
PHL	0.49257	-1.54665	2.65435		
HS	4323.92	4308.46	4329.61		
HL	4323.92	4308.39	4329.57		

REGION 22

	ARRAY 15 SBL	ARRAY 16 SBL	ARRAY 25 SBL	LONGBASE	GOAL
X	57054.15	57054.14	57054.14	57054.15	57054.15
Y	-8587.96	-8587.89	-8588.02	-8587.96	-8587.96
Z	400.00	399.90	399.69	400.00	400.00
THS	0.20118	0.19945	0.17652		
THL	0.20118	0.19942	0.17646		
PHS	-0.52622	-2.61391	1.55241		
PHL	-0.52622	-2.61390	1.55241		
HS	4405.03	4429.00	4416.06		
HL	4405.03	4429.02	4416.12		

REGION 23

	ARRAY 15 SBL	ARRAY 24 SBL	ARRAY 25 SBL	LONGBASE	GOAL
X	53293.11	53293.16	53293.10	53293.11	53293.11
Y	-10707.34	-10707.40	-10707.39	-10707.34	-10707.34
Z	399.99	400.05	399.89	400.00	400.00
THS	0.20470	0.19972	0.17987		
THL	0.20470	0.19973	0.17984		
PHS	-1.55971	0.54136	2.58377		
PHL	-1.55971	0.54138	2.58376		
HS	4332.16	4362.90	4337.36		
HL	4332.16	4362.89	4337.38		

After ERRFIX, and array 15 has almost all of its locational errors corrected. The remaining errors are seen in every array and are the result of the gradient. The LBL poslocs generated in this environment are coincident with the true (goal) posloc. With error free arrays, the gradient does not prevent LONGBASE from accurately determining the targets position.

APPENDIX B: COMPUTER PROGRAMS SIMSTRT AND SETUP

```
      SUBROUTINE SIMSTRT(DVP,EXT,RTIME,RORIGIN)
*
*   THIS SUBROUTINE GENERATES THE TRANSIT TIMES (RTIME) FROM A
*  STARTING POINT ON THE RANGE (RORIGIN) WHICH IS GENERATED BY THE
*  SUBROUTINE POINST.  THIS SUBROUTINE IS TO INITIATE THE NANOOSE RANGE
*  SIMULATION FOR ERROR CORRECTIONS.  THE OUTPUTS CREATED BY THIS
*  PROGRAM SHOULD BE THE SAME AS INPUTS RECEIVED BY THE RANGE (EXCEPT
*  RORIGIN WHICH IS USED TO MONITOR SIMULATION PROGRESS)
*----------------------------------------------------------------------
*  INPUTS:
*     DVP   : THE INDEX OF THE DEPTH VELOCITY PROFILE TO BE USED
*     EXT   : EXTRAPOLATION CONTROL FOR SUBROUTINE VELPRO
*
*  OUTPUTS:
*     RTIME : ARRAY CONTAINING TRANSIT TIMES FROM A LOCATION TO A SENSOR
*             DIMENSIONED AS FOLLOWS: (SENSOR,REGION,I) WHERE I=1-4 THE
*             TIMES FROM THE X,Y,Z,C PHONES RESPECTIVELY
*   RORIGIN : ARRAY CONTAINING THE TRUE LOCATION OF THE TARGET IN RANGE
*             COORDINATES, DIMENSIONED: (REGION,I) I=1-3 FOR X,Y,Z
*                                           J.A. GEMBARSKI
*                                           NPS   03/01/02
*----------------------------------------------------------------------
       DIMENSION XP(3),YP(3),ZP(3),CP(3),A(3),RTIME(0:57,27,4),
     &           AR(3),XPHLOC(7,3),YPHLOC(7,3),ZPHLOC(7,3),CPHLOC(7,3),
     &           RORIGIN(27,3),X(3),ORIGIN(6,3),
     &           L(300),VEL(300),V0(300),V1(300),LM(300),DZ(300),
     &           J(6),LINKS(18),SENSR(7),XS(18,3)
*
       COMMON /SET1/ L,VEL,V0,V1,DZ,LM,M

       INTEGER   SENSR,J,ERR,K,NR,H,KK,N,M,AR,LINKS,NSBL,S,
     &           LP,MP,IEST,EXT,I,DVP
*
       DOUBLE PRECISION    ORIGIN,XS,XPHLOC,YPHLOC,ZPHLOC,CPHLOC,X,A
       DOUBLE PRECISION    XP,YP,ZP,CP,A1,A2,P1,P2,V0,V1,LM,L,VEL,DZ
       DOUBLE PRECISION    TH1,TH2,RTIME,RORIGIN
*
       LOGICAL   TEST
       CHARACTER*15 NAME1
*
       DO 3 I=1,300
          L(I)   = 0.0
          VEL(I) = 0.0
          V0(I)  = 0.0
          V1(I)  = 0.0
          DZ(I)  = 0.0
          LM(I)  = 0.0
     3 CONTINUE
       ERR = 0
      DO 30 S =0,57
        SENSR(1) = S
*
*     FIND THE CORRESPONDING OVERLAP REGIONS AND ADJACENT ARRAYS
*
      CALL FINDOVR(S,J,ERR)
         IF(ERR.EQ.1) GOTO 30

         NR = 1
            DO 10 LP = 1,6
              IF(J(LP).NE.0) NR=NR+1
   10       CONTINUE
         NR = NR-1

      H=1
      K=1
      KK=2
      N=1

   12 IF(J(K).NE.0)THEN
         LINKS(N)=1
         CALL OVERLAP(J(K),AR,ERR)

         DO 13 LP=1,3
            MP=1
   11       IF(SENSR(MP).NE.AR(LP))THEN
                TEST=.TRUE.
                MP=MP+1
            ELSE
                TEST=.FALSE.
            ENDIF
            IF(TEST.AND. MP.LT.KK) GOTO 11

            IF(TEST)THEN
                SENSR(KK)=AR(LP)
                LINKS(N+H)=KK
                KK=KK+1
                H=H+1
            ELSEIF(SENSR(1).NE.AR(LP))THEN
                LINKS(N+H)=MP
                H=H+1
            ENDIF
   13    CONTINUE

         H=1
         K=K+1
         N=N+3
         IF(K.LE.6) GOTO 12
      ENDIF
*
*  FIND THE NUMBER OF SHORT BASE LINE POSLOC NSBL
*
      NSBL = N-1
*
*        LOCATE THE TARGET POSLOC IN EACH CROSSOVER REGION
*
      DO 14 LP=1,NR
         CALL POINTST(J(LP),ORIGIN(LP,1),ORIGIN(LP,2),ORIGIN(LP,3))
         DO 26 H = 1,3
            RORIGIN(J(LP),H) = ORIGIN(LP,H)
   26    CONTINUE
   14 CONTINUE
*
* IDENTIFY AN UNIQUE POSLOC FOR EACH ARRAY IN EACH CROSSOVER REGION
*
      H=1
      DO 20 LP=1,NSBL
         XS(LP,1)=ORIGIN(H,1)
         XS(LP,2)=ORIGIN(H,2)
         XS(LP,3)=ORIGIN(H,3)
         IF(MOD(LP,3).EQ.0) H=H+1
   20 CONTINUE
*
C IDENTIFY THE GLOBAL COORDINATES OF EACH HYDROPHONE OF EACH SENSOR
C              TO CALCULATE THE TRUE TRANSIT TIMES
*
         A2 = 0.0
      DO 21 LP=1,(KK-1)
         CALL ARRAY(SENSR(LP),X(1),X(2),X(3),A(1),A(2),A(3),ERR)
         CALL GLOBE(X,A,XP,YP,ZP,CP)

         DO 22 N=1,3
            XPHLOC(LP,N)=XP(N)
            YPHLOC(LP,N)=YP(N)
            ZPHLOC(LP,N)=ZP(N)
            CPHLOC(LP,N)=CP(N)
   22    CONTINUE
            IF(XP(3).GT.A2)A2 = XP(3)
            IF(YP(3).GT.A2)A2 = YP(3)
            IF(ZP(3).GT.A2)A2 = ZP(3)
            IF(CP(3).GT.A2)A2 = CP(3)
   21 CONTINUE
*
C           RETRIEVE THE VELOCITY PROFILE INFORMATION
*
       IF(EXT.NE.2)EXT = 0
       CALL VELPRO(0,NAME1,DVP,EXT,A2)
*
C    CALCULATE TRANSIT TIMES FROM EACH POSLOC TO EACH HYDROPHONE
C    THE SECOND INDEX ON TIME INDICATES THE HYDROPHONE: 1,2,3,4 FOR
C    X,Y,Z,C RESPECTIVELY (THIS INFORMATION IS NORMALLY SUPPLIED BY
C    THE RANGE)
*
      H = 1
      DO 25 LP=1,NSBL
         A1 = 0.0D0
         A2 = XPHLOC(LINKS(LP),3)
         P1 = DSQRT( (XS(LP,1)-XPHLOC(LINKS(LP),1))**2 +
     +               (XS(LP,2)-XPHLOC(LINKS(LP),2))**2    )
         P2 = XS(LP,3)
         IEST = 0

         CALL RAYFIT1(A1,A2,P1,P2,M,VEL,LM,DZ,V0,V1,
     &                RTIME(SENSR(LINKS(LP)),J(H),1),TH2,TH1,IEST)

         A2 = YPHLOC(LINKS(LP),3)
         P1 = DSQRT( (XS(LP,1)-YPHLOC(LINKS(LP),1))**2 +
     +               (XS(LP,2)-YPHLOC(LINKS(LP),2))**2    )
         IEST = 0

         CALL RAYFIT1(A1,A2,P1,P2,M,VEL,LM,DZ,V0,V1,
     &                RTIME(SENSR(LINKS(LP)),J(H),2),TH2,TH1,IEST)
```

```
      A2 = ZPHLOC(LINKS(LP),3)
      P1 = DSQRT( (XS(LP,1)-ZPHLOC(LINKS(LP),1))**2 +
  +               (XS(LP,2)-ZPHLOC(LINKS(LP),2))**2   )
      IEST = 0

      CALL RAYFIT1(A1,A2,P1,P2,M,VEL,LM,DZ,V0,V1,
  &                RTIME(SENSR(LINKS(LP)),J(H),3),TH2,TH1,IEST)

      A2 = CPHLOC(LINKS(LP),3)
      P1 = DSQRT( (XS(LP,1)-CPHLOC(LINKS(LP),1))**2 +
  +               (XS(LP,2)-CPHLOC(LINKS(LP),2))**2   )
      IEST = 0

      CALL RAYFIT1(A1,A2,P1,P2,M,VEL,LM,DZ,V0,V1,
  &                RTIME(SENSR(LINKS(LP)),J(H),4),TH2,TH1,IEST)

      IF(MOD(LP,3).EQ.0) H= H+1

25 CONTINUE
30 CONTINUE
   RETURN
   END
```

```
************************************************************************
      SUBROUTINE SETUP(DVP,EXT,RTIME,RSHIFT,RSENSR,RLINKS,RJ,RDATA,
     &                 RERRTAB,RXS,RXL,RNR,ROBJ)
*
*    THIS SUBROUTINE TAKES TRANSIT TIMES AND DVP INFORMATION FOR THE
*  RANGE AND GENERATES THE RANGE ARRAYS RSENSR,RLINKS,RJ AND RDATA,
*  RERRTAB FOR INPUT INTO THE RANGE CONTROL PROGRAM.
*-----------------------------------------------------------------------
*  INPUTS:
*     DVP    : THE DVP PROFILE CURRENTLY IN USE
*     EXT    : DVP EXTRAPOLATION CONTROL (2= DO NOT EXTRAPOLATE DVP)
*   RTIME    : ARRAY HOUSING THE TRANSIT TIMES FROM SENSOR NUMBER
*              (FIRST ELEMENT) TO REGION (SECOND ELEMENT) FOR HYDROPHONES
*              X,YZ,C RESPECTIVELY (THIRD ELEMENT)
*   RSHIFT   : ARRAY HOUSING SENSOR REPOSITIONING INFORMATION FOR SENSOR
*              (FIRST ELEMENT) AND DIRECTION X,Y,Z,XTILT,YTILT,ZROT
*              RESPECTIVELY (SECOND ELEMENT)
*  OUTPUTS:
*   RSENSR   : ARRAY HOUSING THE CENTER SENSOR AND ITS TRIPLE OVERLAP
*              SATELLITES (FIRST INDEX SENSOR, SECOND ARRAY SENSR(I))
*   RLINKS   : ARRAY HOUSING THE LINKS(ENSOR) ARRAY FOR EACH SENSOR
*              (FIRST ELEMENT)
*     RJ     : ARRAY HOUSING THE OVERLAP REGIONS ASSOCIATED WITH EACH
*              SENSOR (FIRST INDEX) STORING J(I) ARRAY
*   RDATA    : DATA ARRAY FOR RANGE CONTAINING PHS,PHL,THS,THL,HS,HL
*              (THIRD INDEX) ARRAYS FOR A GIVEN CENTER ARRAY (FIRST
*               INDEX) TO A GIVEN REGION (SECOND INDEX)
*  RERRTAB   : ARRAY CONTAINING THE ERROR ARRAYS THER,PHER,HER,XER,YER,
*              ZER (THIRD INDEX) FOR A GIVEN CENTER ARRAY (FIRST INDEX)
*              TO A GIVEN REGION (SECOND INDEX)
*    RXS     : RANGE SBL ARRAY FOR THE GIVEN CENTER SENSOR (FIRST INDEX)
*              TO A GIVEN REGION (SECOND INDEX)
*    RXL     : RANGE LBL ARRAY FOR THE GIVEN CENTER SENSOR REGION
*              (FIRST INDEX)
*    RNR     : ARRAY CONTIANING NUMBER OF REGIONS ASSOCIATED WITH EACH
*              CENTER SENSOR
*                                                J.A. GEMBARSKI
*                                                NPS   03/01/92
*-----------------------------------------------------------------------
      DIMENSION XP(3),YP(3),ZP(3),CP(3),A(3),HOLD(6),
     &          AR(3),XPHLOC(7,3),YPHLOC(7,3),ZPHLOC(7,3),CPHLOC(7,3),
     &          XS(18,3),XL(6,3),X(3),TIME(18,4),
     &          L(300),VEL(300),V0(300),V1(300),LM(300),
     &          DZ(300),SHIFT(7,6),RSHIFT(0:57,6),LINKS(18),
     &          NEW(3),THL(18),HS(18),SENSR(7),J(6),
     &          THS(18),HL(18),PHS(18),PHL(18),ERROR(7,6),
     &          RTIME(0:57,27,4),RSENSR(0:57,7),RLINKS(0:57,18),
     &          RJ(0:57,6),RDATA(0:57,27,6),RERRTAB(0:57,27,6),
     &          RXS(0:57,27,3),RXL(27,3),RNR(0:57),ROBJ(0:57)
*
      COMMON /SET1/ L,VEL,V0,V1,DZ,LM,M

      INTEGER   SENSR,J,ERR,K,NR,H,KK,N,M,AR,LINKS,NSBL,S,
     +          LP,MP,EXT,I,DVP,RJ,RLINKS,RSENSR,RNR
*
      DOUBLE PRECISION    XS,XL,XPHLOC,YPHLOC,ZPHLOC,CPHLOC,X,A
      DOUBLE PRECISION    XP,YP,ZP,CP,V0,V1,LM,L,VEL,DZ,RTIME,RSHIFT
      DOUBLE PRECISION    NEW,THL,HOLD,RXS,RXL,RERRTAB,RDATA
      DOUBLE PRECISION    TIME,PHS,PHL,HL,HS,THS,SHIFT,ERROR,ROBJ,W
*
      LOGICAL   TEST
      CHARACTER*15 NAME1
*
      ERR = 0
*
C          RETRIEVE THE VELOCITY PROFILE INFORMATION
*
       IF(EXT.NE.2)EXT = 0
       CALL VELPRO(1,NAME1,DVP,EXT,A2)

C        S = 15
       DO 50 S=0,57

        DO 111 I=1,7
           SENSR(I) = -1.0
  111   CONTINUE

        SENSR(1) = S
*
C     FIND THE CORRESPONDING OVERLAP REGIONS AND ADJACENT ARRAYS
C                    IF NONE EXIST SKIP THIS S(ENSOR)
*
      CALL FINDOVR(S,J,ERR)
         IF(ERR.EQ.1) THEN
           ROBJ(S) = -1.0D0
           GOTO 50
         ENDIF

         NR = 1
            DO 10 LP = 1,6
              IF(J(LP).NE.0) NR=NR+1
   10       CONTINUE

         NR = NR-1
         RNR(S) = NR

         DO 51 I =1,6
            RJ(S,I) = J(I)
   51    CONTINUE

      H=1
      K=1
      KK=2
      N=1

   12 IF(J(K).NE.0)THEN
         LINKS(N)=1
         CALL OVERLAP(J(K),AR,ERR)

         DO 13 LP=1,3
            MP=1
   11       IF(SENSR(MP).NE.AR(LP))THEN
                TEST=.TRUE.
                MP=MP+1
            ELSE
                TEST=.FALSE.
            ENDIF
            IF(TEST.AND. MP.LT.KK) GOTO 11

            IF(TEST)THEN
                SENSR(KK)=AR(LP)
                LINKS(N+H)=KK
                KK=KK+1
                H=H+1
            ELSEIF(SENSR(1).NE.AR(LP))THEN
                LINKS(N+H)=MP
                H=H+1
            ENDIF
   13    CONTINUE

         H=1
         K=K+1
         N=N+3
         IF(K.LE.6) GOTO 12
      ENDIF
      DO 52 I = 1,18
         RLINKS(S,I) = LINKS(I)
  52  CONTINUE

      DO 53 I =1,7
         IF(SENSR(I).EQ.-1) GOTO 53
         RSENSR(S,I) = SENSR(I)
  53  CONTINUE

*
*  FIND THE NUMBER OF SHORT BASE LINE POSLOC NSBL
*
      NSBL = N-1
C     WRITE(90,202)
C     DO 40 I=1,18
C        IF(I.LE.6)WRITE(90,203)I,SENSR(I),J(I),LINKS(I),LINKR(I)
C        IF(I.EQ.7)WRITE(90,204)I,SENSR(I),LINKS(I),LINKR(I)
C        IF(I.LE.12.AND.I.GT.7)WRITE(90,205)I,LINKS(I),LINKR(I)
C        IF(I.GT.12)WRITE(90,206)I,LINKS(I)
C  40 CONTINUE
C        CLOSE(UNIT=90)




*
C    ESTABLISH INDUCED ERRORS FOR THE CENTER AND SATELLITE ARRAYS
*
         DO 9 I=1,7
            CALL RERROR(SENSR(I),HOLD,ERR)
            DO 8 H = 1,6
                ERROR(I,H)  = HOLD(H)
    8       CONTINUE
    9    CONTINUE
*
C      RETRIEVE THE ARRAY SHIFT INFORMATION FOR PREVIOUS SHIFTS
*
         DO 7 I=1,7
            DO 6 H = 1,6
               SHIFT(I,H) = RSHIFT(SENSR(I),H)
    6       CONTINUE
    7    CONTINUE
         K = 1
         DO 54 I =1,18
            IF(LINKS(I).EQ.0) GOTO 54
            DO 55 H = 1,4
```

```
               TIME(I,H) = RTIME(SENSR(LINKS(I)),J(K),H)
   55       CONTINUE
            IF(MOD(I,3).EQ.0) K = K+1
   54    CONTINUE
*
*    PERFORM ARRAY SHIFTS FOR ALL SBL POSLOCS
*
      K = 1
      DO 77 I =1,NSBL

      CALL ARRAY(SENSR(LINKS(I)),X(1),X(2),X(3),A(1),A(2),A(3),ERR)

         X(1) = X(1) + ERROR(LINKS(I),1) + SHIFT(LINKS(I),1)
         X(2) = X(2) + ERROR(LINKS(I),2) + SHIFT(LINKS(I),2)
         X(3) = X(3) + ERROR(LINKS(I),3) + SHIFT(LINKS(I),3)

         A(1) = A(1) + ERROR(LINKS(I),4) + SHIFT(LINKS(I),4)
         A(2) = A(2) + ERROR(LINKS(I),5) + SHIFT(LINKS(I),5)
         A(3) = A(3) + ERROR(LINKS(I),6) + SHIFT(LINKS(I),6)

         LP= LINKS(I)
         CALL GLOBE(X,A,XP,YP,ZP,CP)

         DO 32 N=1,3
            XPHLOC(LP,N)=XP(N)
            YPHLOC(LP,N)=YP(N)
            ZPHLOC(LP,N)=ZP(N)
            CPHLOC(LP,N)=CP(N)
   32    CONTINUE

         IF(ERROR(LINKS(I),3).GT.0) THEN
            IF(EXT.NE.2)EXT = 1
            CALL VELPRO(0,NAME1,DVP,EXT,X(3))
         ENDIF

        CALL SHIFTER(X(3),A(1),A(2),A(3),TIME,I,NEW,HS(I),THS(I))

          XS(I,1) = CPHLOC(LINKS(I),1) + NEW(1)
          XS(I,2) = CPHLOC(LINKS(I),2) + NEW(2)
          XS(I,3) = NEW(3)

        IF(MOD(I,3).EQ.0)THEN
          CALL LONG(TIME,LINKS,SENSR,CPHLOC,K,XS,HS,THS,THL,PHS,PHL,HL,
     &             XL)
          K=K+1
        ENDIF

   77   CONTINUE
*
C                  BUILD THE RANGE ARRAYS
*
         K = 1
         DO 60 I = 1,NSBL
            RDATA(SENSR(LINKS(I)),J(K),1) = THS(I)
            RDATA(SENSR(LINKS(I)),J(K),2) = THL(I)
            RDATA(SENSR(LINKS(I)),J(K),3) = PHS(I)
            RDATA(SENSR(LINKS(I)),J(K),4) = PHL(I)
            RDATA(SENSR(LINKS(I)),J(K),5) =  HS(I)
            RDATA(SENSR(LINKS(I)),J(K),6) =  HL(I)
            RXS(SENSR(LINKS(I)),J(K),1)   =  XS(I,1)
            RXS(SENSR(LINKS(I)),J(K),2)   =  XS(I,2)
            RXS(SENSR(LINKS(I)),J(K),3)   =  XS(I,3)
            RERRTAB(SENSR(LINKS(I)),J(K),1) = THL(I) - THS(I)
            RERRTAB(SENSR(LINKS(I)),J(K),2) = PHL(I) - PHS(I)
            RERRTAB(SENSR(LINKS(I)),J(K),3) =  HL(I) -  HS(I)

            RERRTAB(SENSR(LINKS(I)),J(K),4) =  XL(K,1)    -  XS(I,1)
            RERRTAB(SENSR(LINKS(I)),J(K),5) =  XL(K,2)    -  XS(I,2)
            RERRTAB(SENSR(LINKS(I)),J(K),6) =  XL(K,3)    -  XS(I,3)
            IF(MOD(I,3).EQ.0) K = K+1
   60    CONTINUE

         DO 61 K = 1,NR
            RXL(J(K),1)     =  XL(K,1)
            RXL(J(K),2)     =  XL(K,2)
            RXL(J(K),3)     =  XL(K,3)
   61    CONTINUE

         W = 0.0D0
         DO 62 I=1,NR
                 W = W + DSQRT( (XL(I,3)  -XS(3*I-2,3))**2 +
     &                     (HL(3*I-2)-HS(3*I-2))**2 +
     &                     (HS(3*I-2)*(PHL(3*I-2)-PHS(3*I-2)))**2)
   62    CONTINUE
         ROBJ(S) = W/REAL(NR)

   50 CONTINUE
*
      RETURN
      END
```

APPENDIX C: COMPUTER PROGRAMS ERRFIX AND ANNEAL (AND SUPPORT PROGRAMS)

Included with the FORTRAN programs ERRFIX and ANNEAL are the supporting programs NEWTILT and NEWERR. NEWTILT is the tilt correction subcomponent of ERRFIX. NEWERR is a subroutine designed to calculate the new error values from a SHIFT and output them if desired.

```
      SUBROUTINE ERRFIX(TIME,J,LINKS,SENSR,DVP,EXT,NR,ORIGIN,XS,XL,THS,
     &                  THL,PHS,PHL,HS,HL,SHIFT,ERROR)
*
*  THIS SUBROUTINE IS THE ARRAY POSITION (SHIFT 1,2,3) AND ORIENTATION
*  (SHIFT 4,5,6) CORRECTION ALOGRITHM, MINIMIZING SBL AND LBL POSLOCS.
*
*   INPUTS:
*      XS   : SBL POSLOCS
*      XL   : LBL POSLOCS
*      THS  : SBL ELEVATION ANGLES FROM C PHONE
*      THL  : LBL ELEVATION ANGLES FROM C PHONE
*      PHS  : SBL AZIMUTHS
*      PHL  : LBL AZIMUTHS
*      HS   : SBL HORIZONTAL DISTANCES FROM C PHONE
*      HL   : LBL HORIZONTAL DISTANCES FROM C PHONE
*      SHIFT: CORRECTIONS APPLIED TO CENTER ARRAY A.C. PREVIOUSLY
*             X,Y,Z ARE THE 1,2,3 ELEMENTS ;4,5,6 IDENTIFY THE TILTS
*             (XTILT,YTILT,ZROT)
*      ERROR: THE INDUCED ERRORS TO THE SENSOR POSITIONS (GIVING
*             OPERATIONAL POSITIONS) FIRST INDEX CORRESPONDS TO SENSR(I)
*             THE SECOND TO THE TYPE (X,Y,Z,XTILT,YTILT,ZROT)
*   OUTPUTS:
*      SHIFT: X,Y,Z MOVEMENT (FEET) TO BE APPLIED TO ARRAY A.C.
*             XTILT,YTILT,ZROT CORRECTIONS (RADIANS) TO BE APPLIED
*      TIME,J,LINKS,SENSR,DVP,EXT,NR,ORIGIN  : SENSOR ARRAY DATA
*             RECEIVED FROM FORTRAN PROGRAM CONTROL
*      XS,XL,PHS,PHL,THS,THL : UPDATED VALUES OF THESE VECTORS
*                                          J.A. GEMBARSKI
*                                          NPS   03/01/92
*-----------------------------------------------------------------------
*
      DIMENSION XS(18,3),XL(6,3),THS(18),THL(18),PHS(18),
     &          PHL(18),HS(18),HL(18),SHIFT(7,6),SENSR(7),
     &          THER(6),PHER(6),HER(6),XER(6),YER(6),ZER(6),CPH(7,3),
     &          XPH(7,3),YPH(7,3),ZPH(7,3),TIME(18,4),J(6),LINKS(18),
     &          L(300),VEL(300),VC(300),V1(300),DZ(300),LM(300),
     &          ORIGIN(6,3),ERROR(7,6),X(3),A(3),HOLD(7,6)
```

```
*
      COMMON /SET1/ L,VEL,V0,V1,DZ,LM,M
*
      DOUBLE PRECISION  XS,XL,THS,THL,PHS,PHL,HS,HL,TIME,SHIFT,TUNE1,
     &        PHER,THER,HER,XER,YER,ZER,MI,MA,SUM,L,VEL,V0,V1,DZ,LM,
     &        ORIGIN,X,A,CPH,ZFIX,XPH,YPH,ZPH,ERX,ERY,EPS,TUNE,ERROR,
     &        TUNE2,TUNE3,W,OBJ,HOLD
*
      INTEGER J,LINKS,SENSR,NR,EXT,I,M,DVP,C,HC,BAD
*
      LOGICAL CHECK1

      OBJ = 500.0D0
      BAD = 0
      C = 0
      DO 29 I=1,6
         HOLD(1,I) = 0.0D0
   29 CONTINUE
*
C         UPDATE POSITIONS OF C PHONES FROM PREVIOUS SHIFTS
*
      CALL UPDATE(SENSR,DVP,EXT,SHIFT,ERROR,CPH,XPH,YPH,ZPH,X,A)
      CALL NEWERR(C,TIME,J,LINKS,SENSR,ERROR,SHIFT,DVP,EXT,NR,ORIGIN,
     &            THER,PHER,HER,XER,YER,ZER,XS,XL,THS,THL,PHS,PHL,HS,HL)
*
C                     FORM ERROR ARRAYS
*
      DO 30 I=1,NR
         THER(I) = THL((3*I)-2) - THS((3*I)-2)
         PHER(I) = PHL((3*I)-2) - PHS((3*I)-2)
         HER(I) =  HL((3*I)-2) -  HS((3*I)-2)
         XER(I) =  XL(I,1) - XS((3*I)-2,1)
         YER(I) =  XL(I,2) - XS((3*I)-2,2)
         ZER(I) =  XL(I,3) - XS((3*I)-2,3)
   30 CONTINUE

       CHECK1 = .FALSE.
  222  CONTINUE
*----------------------------------------------------------------------
C                   ZROT CORRECTIONS
*----------------------------------------------------------------------
      CALL FIND(PHER,NR,MI,MA)

      IF(MA.LT.0 .OR. MI.GT.0) THEN
        SUM = 0.0
        DO 40 I=1,NR
           SUM = SUM + PHER(I)
   40   CONTINUE

        SHIFT(1,6) = SHIFT(1,6) - (SUM/REAL(NR))

        CALL NEWERR(C,TIME,J,LINKS,SENSR,ERROR,SHIFT,DVP,EXT,NR,ORIGIN,
     &            THER,PHER,HER,XER,YER,ZER,XS,XL,THS,THL,PHS,PHL,HS,HL)
        CHECK1 = .TRUE.
        W = 0.0D0
        DO 80 I =1,NR
           W = W+ DSQRT(ZER(I)**2 + HER(I)**2 + (HS(3*I-2)*PHER(I))**2)
   80   CONTINUE
        W = W/REAL(NR)
        GOTO 222
      ENDIF
```

```
*-----------------------------------------------------------------------
C          DEPTH CORRECTION FROM ELEVATION ANGLE ANALYSIS
*-----------------------------------------------------------------------
      IF(NR.EQ.6)THEN

       ZFIX = 0.0
       CALL FIND(THER,NR,MI,MA)

       IF(MA.LT.0 .OR. MI.GT.0)THEN
        SUM = 0.0
        DO 60 I=1,NR
           SUM = SUM + ZER(I)/(REAL(NR))
   60   CONTINUE
           SHIFT(1,3) = SHIFT(1,3) + SUM

       CALL FIND(ZER,NR,MI,MA)
        CALL NEWERR(C,TIME,J,LINKS,SENSR,ERROR,SHIFT,DVP,EXT,NR,ORIGIN,
     &            THER,PHER,HER,XER,YER,ZER,XS,XL,THS,THL,PHS,PHL,HS,HL)
        CHECK1 = .TRUE.
        W = 0.0D0
        DO 81 I =1,NR
           W = W+ DSQRT(ZER(I)**2 + HER(I)**2 + (HS(3*I-2)*PHER(I))**2)
   81   CONTINUE
        W = W/REAL(NR)
        GOTO 222
       ENDIF

      ENDIF
*-----------------------------------------------------------------------
C           ADJUST XTILT AND YTILT FOR THETA ERRORS
*-----------------------------------------------------------------------
      IF(C.EQ.0.AND.CHECK1 .OR. CHECK1) THEN
        CALL NEWTILT(SENSR,DVP,EXT,PHL,THL,PHS,THS,NR,SHIFT,ERX,ERY,
     &             ERROR)

        SHIFT(1,4) = SHIFT(1,4) + ERX
        SHIFT(1,5) = SHIFT(1,5) + ERY
        CALL NEWERR(C,TIME,J,LINKS,SENSR,ERROR,SHIFT,DVP,EXT,NR,ORIGIN,
     &            THER,PHER,HER,XER,YER,ZER,XS,XL,THS,THL,PHS,PHL,HS,HL)
        CHECK1 = .FALSE.
        W = 0.0D0
        DO 82 I =1,NR
           W = W+ DSQRT(ZER(I)**2 + HER(I)**2 + (HS(3*I-2)*PHER(I))**2)
   82   CONTINUE
        W = W/REAL(NR)
        IF(W.LE.OBJ) THEN
          OBJ = W
          DO 182 I=1,6
             HC = C
             HOLD(1,I) = SHIFT(1,I)
  182     CONTINUE
          BAD = 0
        ELSE
          BAD = BAD + 1
        ENDIF
        IF(BAD.GT.5) GOTO 444
      ENDIF

      TUNE = (C+1.0)/6.0
      IF(C.GE.15) TUNE = 3.0D0
```

```
*----------------------------------------------------------------------
C             LOOK FOR CONSISTENT Y CORRECTIONS
*----------------------------------------------------------------------
      CALL FIND(YER,NR,MI,MA)
      IF(MA.LT.0 .OR. MI.GT.0)THEN
        SUM = 0.0
        DO 63 I =1,NR
           SUM = SUM + YER(I)/(REAL(NR))
   63   CONTINUE
              TUNE1 = TUNE
           SHIFT(1,2) = SHIFT(1,2) + SUM*TUNE1
        CALL NEWERR(C,TIME,J,LINKS,SENSR,ERROR,SHIFT,DVP,EXT,NR,ORIGIN,
     &            THER,PHER,HER,XER,YER,ZER,XS,XL,THS,THL,PHS,PHL,HS,HL)
        CHECK1 = .TRUE.
        W = 0.0D0
        DO 83 I =1,NR
           W = W+ DSQRT(ZER(I)**2 + HER(I)**2 + (HS(3*I-2)*PHER(I))**2)
   83   CONTINUE
        W = W/REAL(NR)
        GOTO 222
      ENDIF

*----------------------------------------------------------------------
C             LOOK FOR CONSISTENT X CORRECTIONS
*----------------------------------------------------------------------
      CALL FIND(XER,NR,MI,MA)
      IF(MA.LT.0 .OR. MI.GT.0)THEN
        SUM = 0.0
        DO 64 I =1,NR
           SUM = SUM + XER(I)/(REAL(NR))
   64   CONTINUE
              TUNE2 = TUNE
           SHIFT(1,1) = SHIFT(1,1) + SUM * TUNE2
        CALL NEWERR(C,TIME,J,LINKS,SENSR,ERROR,SHIFT,DVP,EXT,NR,ORIGIN,
     &            THER,PHER,HER,XER,YER,ZER,XS,XL,THS,THL,PHS,PHL,HS,HL)
        CHECK1 = .TRUE.
        W = 0.0D0
        DO 84 I =1,NR
           W = W+ DSQRT(ZER(I)**2 + HER(I)**2 + (HS(3*I-2)*PHER(I))**2)
   84   CONTINUE
        W = W/REAL(NR)
        GOTO 222
      ENDIF

*----------------------------------------------------------------------
C              LOOK FOR CONSISTENT Z CORRECTIONS
*----------------------------------------------------------------------
      CALL FIND(ZER,NR,MI,MA)
      IF(MA.LT.0 .OR. MI.GT.0)THEN
         SUM = 0.0
         DO 65 I =1,NR
            SUM = SUM + ZER(I)/(REAL(NR))
   65    CONTINUE
              TUNE3 = TUNE*1.5
            SHIFT(1,3) = SHIFT(1,3) + SUM * TUNE3
        CALL NEWERR(C,TIME,J,LINKS,SENSR,ERROR,SHIFT,DVP,EXT,NR,ORIGIN,
     &            THER,PHER,HER,XER,YER,ZER,XS,XL,THS,THL,PHS,PHL,HS,HL)
        CHECK1 = .TRUE.
        W = 0.0D0
        DO 85 I =1,NR
           W = W+ DSQRT(ZER(I)**2 + HER(I)**2 + (HS(3*I-2)*PHER(I))**2)
   85   CONTINUE
        W = W/REAL(NR)
        GOTO 222
      ENDIF

*------------------------------------------------------------------------
C                  LOOK FOR Y CORRECTIONS
*------------------------------------------------------------------------
        SUM = 0.0
        DO 66 I =1,NR
           SUM = SUM + YER(I)/(REAL(NR))
   66   CONTINUE
              TUNE1 = TUNE
           SHIFT(1,2) = SHIFT(1,2) + SUM*TUNE1
        CALL NEWERR(C,TIME,J,LINKS,SENSR,ERROR,SHIFT,DVP,EXT,NR,ORIGIN,
     &           THER,PHER,HER,XER,YER,ZER,XS,XL,THS,THL,PHS,PHL,HS,HL)
        W = 0.0D0
        DO 86 I =1,NR
           W = W+ DSQRT(ZER(I)**2 + HER(I)**2 + (HS(3*I-2)*PHER(I))**2)
   86   CONTINUE
        W = W/REAL(NR)

*------------------------------------------------------------------------
C                  LOOK FOR X CORRECTIONS
*------------------------------------------------------------------------
        SUM = 0.0
        DO 67 I =1,NR
           SUM = SUM + XER(I)/(REAL(NR))
   67   CONTINUE
              TUNE2 = TUNE
           SHIFT(1,1) = SHIFT(1,1) + SUM * TUNE2
        CALL NEWERR(C,TIME,J,LINKS,SENSR,ERROR,SHIFT,DVP,EXT,NR,ORIGIN,
     &           THER,PHER,HER,XER,YER,ZER,XS,XL,THS,THL,PHS,PHL,HS,HL)
        W = 0.0D0
        DO 87 I =1,NR
           W = W+ DSQRT(ZER(I)**2 + HER(I)**2 + (HS(3*I-2)*PHER(I))**2)
   87   CONTINUE
        W = W/REAL(NR)
*------------------------------------------------------------------------
C                 LOOK FOR Z CORRECTIONS
*------------------------------------------------------------------------
         SUM = 0.0
         DO 68 I =1,NR
            SUM = SUM + ZER(I)/(REAL(NR))
   68    CONTINUE
              TUNE3 = TUNE
            SHIFT(1,3) = SHIFT(1,3) + SUM * TUNE3
        CALL NEWERR(C,TIME,J,LINKS,SENSR,ERROR,SHIFT,DVP,EXT,NR,ORIGIN,
     &           THER,PHER,HER,XER,YER,ZER,XS,XL,THS,THL,PHS,PHL,HS,HL)
        CHECK1 = .TRUE.
        W = 0.0D0
        DO 88 I =1,NR
           W = W+ DSQRT(ZER(I)**2 + HER(I)**2 + (HS(3*I-2)*PHER(I))**2)
   88   CONTINUE
        W = W/REAL(NR)

      C = C + 1
      IF(C.LT.60 .AND. BAD.LE.5)GOTO 222

  444 WRITE(92,*)
      WRITE(92,*)'C',C,'BAD',BAD
      WRITE(92,500)SHIFT(1,1),SHIFT(1,2),SHIFT(1,3)
      WRITE(92,501)SHIFT(1,4),SHIFT(1,5),SHIFT(1,6)
      WRITE(92,*) HC
      WRITE(92,502)HOLD(1,1),HOLD(1,2),HOLD(1,3)
      WRITE(92,503)HOLD(1,4),HOLD(1,5),HOLD(1,6)
      DO 555 I=1,6
         SHIFT(1,I) = HOLD(1,I)
  555 CONTINUE

  500 FORMAT(10X,'PS11 ',2X,F9.5,2X,'PS12 ',F9.5,2X,'PS13 ',F9.5)
  501 FORMAT(10X,'AS11 ',2X,F9.5,2X,'AS12 ',F9.5,2X,'AS13 ',F9.5)
  502 FORMAT(10X,'HS11 ',2X,F9.5,2X,'HS12 ',F9.5,2X,'HS13 ',F9.5)
  503 FORMAT(10X,'HS11 ',2X,F9.5,2X,'HS12 ',F9.5,2X,'HS13 ',F9.5)


      RETURN
      END







***************************************************************************
      SUBROUTINE ANNEAL(TIME,J,LINKS,SENSR,DVP,EXT,NR,ORIGIN,XS,XL,THS,
     &                  THL,PHS,PHL,HS,HL,SHIFT,ERROR,WMIN)
*
*  THIS SUBROUTINE IS THE ARRAY POSITION (SHIFT 1,2,3) AND ORIENTATION
*  (SHIFT 4,5,6) CORRECTION ALOGRITHM, MINIMIZING SBL AND LBL POSLOCS
*  USING SIMULATED ANNEALING.
*
*   INPUTS:
*      XS   : SBL POSLOCS
*      XL   : LBL POSLOCS
*      THS  : SBL ELEVATION ANGLES FROM C PHONE
*      THL  : LBL ELEVATION ANGLES FROM C PHONE
*      PHS  : SBL AZIMUTHS
*      PHL  : LBL AZIMUTHS
*      HS   : SBL HORIZONTAL DISTANCES FROM C PHONE
*      HL   : LBL HORIZONTAL DISTANCES FROM C PHONE
*      SHIFT: CORRECTIONS APPLIED TO CENTER ARRAY A.C. PREVIOUSLY
*             X,Y,Z ARE THE 1,2,3 ELEMENTS ;4,5,6 IDENTIFY THE TILTS
*             (XTILT,YTILT,ZROT)
*      ERROR: THE INDUCED ERRORS TO THE SENSOR POSITIONS (GIVING
*             OPERATIONAL POSITIONS) FIRST INDEX CORRESPONDS TO SENSR(I)
*             THE SECOND TO THE TYPE (X,Y,Z,XTILT,YTILT,ZROT)
*      WMIN : INITIAL GUESS TO THE MINIMUM OBJECTIVE FUNCTION VALUE
*      TIME,J,LINKS,SENSR,DVP,EXT,NR,ORIGIN  : SENSOR ARRAY DATA
*             RECEIVED FROM FORTRAN PROGRAM CONTROL
*
*   OUTPUTS:
*      SHIFT: X,Y,Z MOVEMENT (FEET) TO BE APPLIED TO ARRAY A.C.
*             XTILT,YTILT,ZROT CORRECTIONS (RADIANS) TO BE APPLIED
*      XS,XL,PHS,PHL,THS,THL : UPDATED VALUES OF THESE VECTORS
*                                                J.A. GEMBARSKI
*                                                NPS   03/01/92
*--------------------------------------------------------------------------
*
      DIMENSION XS(18,3),XL(6,3),THS(18),THL(18),PHS(18),
     &          PHL(18),HS(18),HL(18),SHIFT(7,6),SENSR(7),SMHOLD(7),
     &          THER(6),PHER(6),HER(6),XER(6),YER(6),ZER(6),CPH(7,3),
     &          XPH(7,3),YPH(7,3),ZPH(7,3),TIME(18,4),J(6),LINKS(18),
     &          L(300),VEL(300),V0(300),V1(300),DZ(300),LM(300),
     &          ORIGIN(6,3),ERROR(7,6),X(3),A(3),HOLD(7,6),Y(6),U(6)
*
      COMMON /SET1/ L,VEL,V0,V1,DZ,LM,M
*
      DOUBLE PRECISION  XS,XL,THS,THL,PHS,PHL,HS,HL,TIME,SHIFT,
     &        PHER,THER,HER,XER,YER,ZER,SUM,L,VEL,V0,V1,DZ,LM,
     &        ORIGIN,X,A,CPH,XPH,YPH,ZPH,EPS,ERROR,SMALL,SMHOLD,
     &        HOLD,W,B,DELR1,G,WMIN,W0,W1,U,PI,P,DELR2,ALPHA,DEL
      REAL    V,Y
*
      INTEGER J,LINKS,SENSR,NR,EXT,I,M,DVP,C,BAD,IX,NO,BAD1
*
      LOGICAL CHECK1,CHECK2,JUMP100,JUMP200
*
C     PARAMETER DEFINITION
*       IX     SEED FOR THE RANDOM NUMBER GENERATORS
*       BAD    NUMBER OF SELECTED 'WRONG' STEPS MADE
*       BAD1   NUMBER OF STEPS TRIED WITHOUT IMPROVING ON THE SMALLEST
*              OBJECTIVE FUNCTION FOUND SO FAR
*       ALPHA  USED IN DETERMINING A NEW OBJECTIVE FUNCTION VALUE IF
*              WMIN EQUALED OF SURPASSED
*       B      MULTIPLIER FOR DETERMINATION OF PROBABILITY OF ACCEPTING
*              'WRONG' STEP
*       G      EXPONENT FOR INCREMENTAL OBJECTIVE FUNCTION CHANGE  FOR
*              DETERMINATION OF PROBABILITY OF ACCEPTING 'WRONG' STEP
*       DELR1  MULTIPLIER FOR POSITIONAL STEP SIZE
*       DELR2  MULTIPLIER FOR ANGULAR STEP SIZE

      NO = 6
      PI = DACOS(-1.0D0)
      EPS = 1.0D-4
      BAD = 0
      BAD1 = 0
      IX = 16806
      JUMP100 = .FALSE.
      JUMP200 = .FALSE.
      ALPHA = 0.4D0
      B =  NR*2.5 + ABS(NR-3)
        DELR1 = 0.011D0
        DELR2 = 7.7D-6
      G = 1.0D0
C     WMIN = 0.0D0
      C = 0
*
*
C         UPDATE POSITIONS OF C PHONES FROM PREVIOUS SHIFTS
*
      CALL UPDATE(SENSR,DVP,EXT,SHIFT,ERROR,CPH,XPH,YPH,ZPH,X,A)
        CALL NEWERR(C,TIME,J,LINKS,SENSR,ERROR,SHIFT,DVP,EXT,NR,ORIGIN,
     &            THER,PHER,HER,XER,YER,ZER,XS,XL,THS,THL,PHS,PHL,HS,HL)

*
C                     FORM ERROR ARRAYS
*
      DO 10 I=1,NR
         THER(I) = THL((3*I)-2) - THS((3*I)-2)
         PHER(I) = PHL((3*I)-2) - PHS((3*I)-2)
         HER(I) =  HL((3*I)-2) -  HS((3*I)-2)
         XER(I) =  XL(I,1) - XS((3*I)-2,1)
         YER(I) =  XL(I,2) - XS((3*I)-2,2)
         ZER(I) =  XL(I,3) - XS((3*I)-2,3)
   10 CONTINUE
          PRINT*,SHIFT(1,1),SHIFT(1,2),SHIFT(1,3)
          PRINT*,SHIFT(1,4),SHIFT(1,5),SHIFT(1,6)


*----------------------------------------------------------------------
C                      BUILD HOLD ARRAY
*----------------------------------------------------------------------

        DO  9 I = 1,7
           DO 8 K=1,6
              HOLD(I,K) = SHIFT(I,K)
    8      CONTINUE
    9   CONTINUE
*--------------------------------------------------------------------
C  CALCULATE NEW ERRORS TERMS USING A WEIGHTED CONSENSUS POSITION
*--------------------------------------------------------------------
C       DO 88 I = 1,NR
C       IF(DABS(XS(3*I-1,2)-XS(3*I,2)).LT.0.01) THEN
C          YER(I) = (XS(3*I-1,2)+XS(3*I,2))/2.0 - XS(3*I-2,2)
C          CHECK1 = .TRUE.
C        ELSE
C          CHECK1 = .FALSE.
C          YER(I) = (XS(3*I-1,2)+XS(3*I,2)+XL(I,2))/3.0 - XS(3*I-2,2)
C        ENDIFC

C        IF(DABS(XS(3*I-1,1)-XS(3*I,1)).LT.0.01) THEN
C          CHECK2 = .TRUE.
C          XER(I) = (XS(3*I-1,1)+XS(3*I,1))/2.0 - XS(3*I-2,1)
C        ELSE
C          CHECK2 = .FALSE.
C          XER(I) = (XS(3*I-1,1)+XS(3*I,1)+XL(I,1))/3.0 - XS(3*I-2,1)
C        ENDIF
C        IF(CHECK1 .AND. CHECK2) THEN
C           HER(I) = DSQRT(XER(I)**2 + YER(I)**2)
C        ENDIF

C        IF(DABS(XS(3*I-1,3)-XS(3*I,3)).LT.0.01) THEN
C          ZER(I) = (XS(3*I-1,3)+XS(3*I,3))/2.0 - XS(3*I-2,3)
C        ELSE
C          ZER(I) = (XS(3*I-1,3)+XS(3*I,3)+XL(I,3))/3.0 - XS(3*I-2,3)
C        ENDIF
C   88  CONTINUE

        W = 0.0D0
        DO 80 I =1,NR
           W = W+ DSQRT(ZER(I)**2+HER(I)**2+(HS(3*I-2)*PHER(I))**2)
   80   CONTINUE
           WO = W/REAL(NR)
*-----------------------------------------------------------------------
C   CALCULATE NEW WMIN IF THE OBJECTIVE FUNCTION IS LESS THAN OR EQUAL
C     TO THE INITIAL WMIN
*-----------------------------------------------------------------------
        DEL = WO-WMIN
        IF(DEL.LT.0.0) THEN
          WMIN = WO + ALPHA*DEL
          IF(WMIN.LT.0.0) WMIN = 0.0
```

```
C         WRITE(91,*)'NEW WMIN',WMIN
        ELSEIF(DEL.LE.EPS .AND. DEL.GT.0.0) THEN
          GOTO 555
        ENDIF
        SMALL = WO*2.0

*-----------------------------------------------------------------------
C               CHOOSE A RANDOM DIRECTION TO WALK
*-----------------------------------------------------------------------
   60 IF(BAD.GT.30) GOTO 555
C       PRINT*,'WO',WO
C       PRINT*,'BAD',BAD
        CALL LNORM(IX,Y,NO,1,0)
        SUM = 0.0

        DO 12 I = 1,6
           SUM = SUM + ABS(Y(I))**2.0D0
   12   CONTINUE

        DO 13 I=1,6
           U(I) = Y(I)/SQRT(SUM)
   13   CONTINUE

        DO 14 I = 1,2
           HOLD(1,I) = SHIFT(1,I) + DELR1*U(I)
   14   CONTINUE
           HOLD(1,3) = SHIFT(1,3) + 1.5*DELR1*U(3)
        DO 24 I = 4,5
           HOLD(1,I) = SHIFT(1,I) + DELR2*U(I)
   24   CONTINUE
           HOLD(1,6) = SHIFT(1,6) + 0.5*DELR2*U(6)

*----------------------------------------------------------------------
C                 CHECK FOR VALID STEPS
*----------------------------------------------------------------------

      IF(HOLD(1,4).LT. (-PI/2.0) .OR. HOLD(1,4).GT.(PI/2.0)) GOTO 60
      IF(HOLD(1,5).LT. (-PI/2.0) .OR. HOLD(1,5).GT.(PI/2.0)) GOTO 60
      IF(HOLD(1,6).LT. (-PI) .OR. HOLD(1,6).GT.(PI)) GOTO 60

        CALL NEWERR(C,TIME,J,LINKS,SENSR,ERROR,HOLD,DVP,EXT,NR,ORIGIN,
     &            THER,PHER,HER,XER,YER,ZER,XS,XL,THS,THL,PHS,PHL,HS,HL)

*-------------------------------------------------------------------
C  CALCULATE NEW ERRORS TERMS USING A WEIGHTED CONSENSUS POSITION
*-------------------------------------------------------------------
C       DO 89 I = 1,NR
C        IF(DABS(XS(3*I-1,2)-XS(3*I,2)).LT.0.01) THEN
C          CHECK1 = .TRUE.
C          YER(I) = (XS(3*I-1,2)+XS(3*I,2))/2.0 - XS(3*I-2,2)
C        ELSE
C          CHECK1 = .FALSE.
C          YER(I) = (XS(3*I-1,2)+XS(3*I,2)+XL(I,2))/3.0 - XS(3*I-2,2)
C        ENDIF

C        IF(DABS(XS(3*I-1,1)-XS(3*I,1)).LT.0.01) THEN
C          CHECK2 = .TRUE.
C          XER(I) = (XS(3*I-1,1)+XS(3*I,1))/2.0 - XS(3*I-2,1)
C        ELSE
C          CHECK2 = .FALSE.
C          XER(I) = (XS(3*I-1,1)+XS(3*I,1)+XL(I,1))/3.0 - XS(3*I-2,1)
```

```
C      ENDIF
C      IF(CHECK1 .AND. CHECK2) THEN
C         HER(I) = DSQRT(XER(I)**2 + YER(I)**2)
C      ENDIF

C      IF(DABS(XS(3*I-1,3)-XS(3*I,3)).LT.0.01) THEN
C        ZER(I) = (XS(3*I-1,3)+XS(3*I,3))/2.0 - XS(3*I-2,3)
C      ELSE
C        ZER(I) = (XS(3*I-1,3)+XS(3*I,3)+XL(I,3))/3.0 - XS(3*I-2,3)
C      ENDIF
C   89 CONTINUE
*---------------------------------------------------------------------
C   CHANGE THE STEP SIZING AND PROBABILITY CALCULATING PARAMETERS IF
C             NOT IMPROVING THE OBJECTIVE FUNCTION
*---------------------------------------------------------------------
      W = 0.0D0
      DO 15 I =1,NR
         W = W+ DSQRT(ZER(I)**2 + HER(I)**2 + (HS(3*I-2)*PHER(I))**2)
   15 CONTINUE
         W1 = W/REAL(NR)

      DELW = W1 - W0
      IF(W0.LE.SMALL .OR. W1.LE.SMALL) THEN
        BAD1 = 0

        IF(JUMP100 .AND. .NOT.JUMP200) THEN
         DELR1 = DELR1/1.5
         DELR2 = DELR2/1.5
         JUMP100 = .FALSE.
        ENDIF

        IF(JUMP100 .AND. JUMP200) THEN
         B = B/1.25
         JUMP100 = .FALSE.
         JUMP200 = .FALSE.
        ENDIF

        IF(W0.LE.W1) THEN
          DO 18 I=1,NR
             SMHOLD(I) = SHIFT(1,I)
   18     CONTINUE
          SMALL = W0
        ELSE
          DO 19 I=1,NR
             SMHOLD(I) = HOLD(1,I)
   19     CONTINUE
          SMALL = W1
        ENDIF
      ELSE
        BAD1 = BAD1 + 1
      ENDIF
      IF(BAD1.EQ.100) THEN
         DELR1 = DELR1*1.5
         DELR2 = DELR2*1.5
         JUMP100 = .TRUE.
         PRINT*,'JUMP100'
      ELSEIF(BAD1.EQ.200) THEN
         B = B*1.25
         JUMP200 = .TRUE.
         PRINT*,'JUMP200'
      ENDIF
```

```
           IF(BAD1 .EQ. 2500) GOTO 555

*----------------------------------------------------------------------
C             DETERMINE WHETHER OR NOT TO TAKE THE STEP
*----------------------------------------------------------------------
      IF(W1.LE.WO) THEN

        DO 16 I = 1,6
           SHIFT(1,I) = HOLD(1,I)
   16   CONTINUE


         WO = W1

         DEL = WO-WMIN
         IF(DEL.LT.0.0) THEN
           WMIN = WO + ALPHA*DEL
           IF(WMIN.LT.0.0) WMIN = 0.0
         ELSEIF(DEL.LE.EPS .AND. DEL.GT.0.0) THEN
           GOTO 555
         ENDIF

           C = C + 1
           BAD = 0
           GOTO 60

      ELSE
           BAD = BAD +1
C          G = 1.0 - 2.0*WO
           IF((-B*DELW/(WO**G)).LT.-50.0) THEN
              P = 0.0
           ELSE
              P = EXP(-B*DELW/(WO**G))
           ENDIF

           CALL LRND(IX,V,1,1,0)

           IF(V.GE.P) THEN
C            WRITE(91,*)'TAKE BAD STEP'
             C = C + 1
             GOTO 60
           ELSE
C            WRITE(91,*)'TAKE BAD STEP'
             DO 17 I = 1,6
               SHIFT(1,I) = HOLD(1,I)
   17        CONTINUE
             WO = W1
             GOTO 60
           ENDIF
      ENDIF

  555 IF(BAD1.EQ.2500) THEN
        DO 20 I =1,NR
           SHIFT(1,I) = SMHOLD(I)
   20   CONTINUE
      ENDIF
      WRITE(92,*)'C',C,' BAD1 ',BAD1
      WRITE(92,*)
      WRITE(92,500)SHIFT(1,1),SHIFT(1,2),SHIFT(1,3)
      WRITE(92,501)SHIFT(1,4),SHIFT(1,5),SHIFT(1,6)
      WRITE(92,*)
      WRITE(92,*) 'SMALLEST ',SMALL
      WRITE(92,500)SMHOLD(1),SMHOLD(2),SMHOLD(3)
      WRITE(92,501)SMHOLD(4),SMHOLD(5),SMHOLD(6)
  500 FORMAT(10X,'PS11 ',2X,F9.5,2X,'PS12 ',F9.5,2X,'PS13 ',F9.5)
  501 FORMAT(10X,'AS11 ',2X,F9.5,2X,'AS12 ',F9.5,2X,'AS13 ',F9.5)

      RETURN
      END
```

```
***********************************************************************
      SUBROUTINE NEWTILT(SENSR,DVP,EXT,PHL,THL,PHS,THS,NR,SHIFT,ERX,
     &                   ERY,ERROR)

*  SUBROUTINE TO ERRFIX
*  FIND NEW POSITIONS OF THE X AND Y HYDROPHONES BASED ON TWO
*  TILT ERRORS, THOSE CLOSEST TO THE Y AND X AXIS, THIS IS DONE
*  BY FITTING A PLANE THROUGH THE NEW POINTS ALONG THE CHOSEN
*  AZIMUTHS AND THEN BACKING OUT THE X,Y HYDROPHONE POSITIONS
*
*  INPUTS:
*     ERROR : ERROR ARRAY CONTAINING INDUCED ERRORS TO THE CENTER AND
*             SATELLITE SENSORS-FIRST ELEMENT CONTROLS WHICH SENSOR
*             THE SECOND CONTROLS CORRECTION (1-3; X,Y,Z 4-6; X,Y,Z
*             TILTS)
*     SHIFT : ARRAY CONTAINING CORRECTIONS TO THE CENTER AND
*             SATELLITE SENSORS-FIRST ELEMENT CONTROLS WHICH SENSOR
*             THE SECOND CONTROLS CORRECTION (1-3; X,Y,Z 4-6; X,Y,Z
*             TILTS)
*     SENSR : VECTOR DEFINING SENSOR ARRAY NUMBERS
*     DVP,EXT: DVP IDENTIFIER (FROM CONTROL PROGRAM) AND EXTRAPOLATION
*             CONTROL
*     PHL,PHS,THL,THS :  LBL AND SBL POSLOC INFORMATION
*     NR    :  NUMBER OF OVERLAP REGIONS FOR THE INTERIOR ARRAY OF
*              INTEREST
*  OUTPUTS:
*      ERX   : CORRECTION TERM FOR XTILT ANGLE (RADIANS)
*      ERY   : CORRECTION TERM FOR YTILT ANGLE (RADIANS)
*
*----------------------------------------------------------------------
      DIMENSION PHL(18),SHIFT(7,6),X(3),Y(3),THL(18),
     &          CO(3),XPH(7,3),YPH(7,3),ZPH(7,3),CPH(7,3),PHS(18),
     &          THS(18),ERROR(7,6),SENSR(7),PHER(6)
      DOUBLE PRECISION PHL,PI,ZROT,R,HOLD90,SHIFT,HOLDO,D,ERX,
     &                 CO,XPH,YPH,ZPH,CPH,X,Y,YTILTL,XTILTL,A,B,C,E,ERY,
     &                 XHX,YHX,ZHX,XHY,YHY,ZHY,THL,PHS,THS,XTILTS,
     &                 YTILTS,ERROR,PHER,MA,MI
      INTEGER  MARK90,I,NR,MARK0,EXT,SENSR,DVP,K

      CALL UPDATE(SENSR,DVP,EXT,SHIFT,ERROR,CPH,XPH,YPH,ZPH,X,Y)
      ZROT = Y(3)
      D = 30.0D0
      PI = DACOS(-1.0D0)
      MARK90 = 0
      HOLD90 = 5.0*PI/6.0
      MARK0  = 0
      HOLD0  = 5.0*PI/6.0
*-------------------------------------------------------------------
C       FIND POSLOC REGIONS CLOSEST TO 0 AND 90 DEGREES (RANGE)
*-------------------------------------------------------------------
      DO 10 I=1,NR
         IF(DABS(PHL(3*I-2)+ZROT).LE.PI)THEN
           R = PHL(3*I-2) + ZROT
         ELSEIF(ZROT.GT.0.0)THEN
           R = PHL(3*I-2) + ZROT - 2.0*PI
         ELSE
           R = 2.0*PI + PHL(3*I-2) + ZROT
         ENDIF

         IF(DABS(R-PI/2.0).LT.HOLD90)THEN
           HOLD90 = DABS(R-PI/2.0)
           MARK90 = I
         ENDIF
         IF(DABS(R).LT.HOLD0)THEN
           HOLD0 = DABS(R)
           MARK0 = I
         ENDIF

   10 CONTINUE

       IF(MARK90.EQ.MARK0 .AND. NR.GT.1)THEN
         IF((NR-MARK90).GT.0)THEN
           MARK0 = MARK90 + 1
         ELSE
           MARK0 = MARK90 - 1
         ENDIF
       ELSEIF(NR.EQ.1)THEN
         PRINT*,'NOT ENOUGH REGIONS TO FORM A PLANE'
         PRINT*,'RETURN FROM NEWTILT'
         RETURN
       ENDIF

       IF(MARK90.EQ.0)THEN
          PRINT*,'ERROR IN Y AXIS LOCATION'
          STOP
       ELSEIF(MARK0.EQ.0)THEN
          PRINT*,'ERROR IN X AXIS LOCATION'
          STOP
       ENDIF

       DO 12 I=1,3
         CO(I) = CPH(1,I)
   12  CONTINUE
*
*           FORM THE PLANE THROUGH THE LBL ANGLES
*
```

```
        X(1) =  D*DCOS(PHL(3*MARK90-2)) + CO(1)
        X(2) =  D*DSIN(PHL(3*MARK90-2)) + CO(2)
        X(3) = -D*DTAN(THL(3*MARK90-2)) + CO(3)
        Y(1) =  D*DCOS(PHL(3*MARKO -2)) + CO(1)
        Y(2) =  D*DSIN(PHL(3*MARKO -2)) + CO(2)
        Y(3) = -D*DTAN(THL(3*MARKO -2)) + CO(3)

      CALL PLANE(X,Y,CO,A,B,C,E)

      XHX = D*DCOS(-ZROT) + CO(1)
      YHX = D*DSIN(-ZROT) + CO(2)
      ZHX = CO(3) - (-E-A*XHX-B*YHX)/C
      XTILTL = DASIN(ZHX/D)

      XHY = D*DCOS(-ZROT+PI/2.0) + CO(1)
      YHY = D*DSIN(-ZROT+PI/2.0) + CO(2)
      ZHY = CO(3) - (-E-A*XHY-B*YHY)/C
      YTILTL = DASIN(ZHY/D)
*
*         FORM THE PLANE THROUGH THE SBL ANGLES
*
        X(1) =  D*DCOS(PHS(3*MARK90-2)) + CO(1)
        X(2) =  D*DSIN(PHS(3*MARK90-2)) + CO(2)
        X(3) = -D*DTAN(THS(3*MARK90-2)) + CO(3)
        Y(1) =  D*DCOS(PHS(3*MARKO -2)) + CO(1)
        Y(2) =  D*DSIN(PHS(3*MARKO -2)) + CO(2)
        Y(3) = -D*DTAN(THS(3*MARKO -2)) + CO(3)

      CALL PLANE(X,Y,CO,A,B,C,E)

      XHX = D*DCOS(-ZROT) + CO(1)
      YHX = D*DSIN(-ZROT) + CO(2)
      ZHX = CO(3) - (-E-A*XHX-B*YHX)/C
      XTILTS = DASIN(ZHX/D)

      XHY = D*DCOS(-ZROT+PI/2.0) + CO(1)
      YHY = D*DSIN(-ZROT+PI/2.0) + CO(2)
      ZHY = CO(3) - (-E-A*XHY-B*YHY)/C
      YTILTS = DASIN(ZHY/D)

      ERX = (XTILTL-XTILTS)
      ERY = (YTILTL-YTILTS)

      RETURN
      END
```

```
************************************************************************
      SUBROUTINE NEWERR(C,TIME,J,LINKS,SENSR,ERROR,SHIFT,DVP,EXT,NR,
     &                  ORIGIN,THER,PHER,HER,XER,YER,ZER,XS,XL,THS,THL,
     &                  PHS,PHL,HS,HL)
*
*   THIS SUBROUTINE PERFORMS A SENSOR SHIFT FROM THE SHIFT ARRAY
*   INFORMATION AND THEN CALCULATES A NEW LONGBASE POSLOC AND ERROR
*   TABLE. CALLED FROM ERRFIX.
*  INPUTS:
*     C     : OUTPUT CONTROL NUMBER, NUMBER OF ITERATIONS THROUGH
*             ERRFIX SUBROUTINE
*     ERROR : ERROR ARRAY CONTAINING INDUCED ERRORS TO THE CENTER AND
*             SATELLITE SENSORS-FIRST ELEMENT CONTROLS WHICH SENSOR
*             THE SECOND CONTROLS CORRECTION (1-3; X,Y,Z 4-6· X,Y,Z
*             TILTS)
*     SHIFT : ARRAY CONTAINING CORRECTIONS TO THE CENTER AND
*             SATELLITE SENSORS-FIRST ELEMENT CONTROLS WHICH SENSOR
*             THE SECOND CONTROLS CORRECTION (1-3; X,Y,Z 4-6; X,Y,Z
*             TILTS)
*     TIME, LINKS, SENSR, J:  INDIVIDUAL ARRAY VECTORS AS DEFINED IN\
*             SUBROUTINE SETUP FOR THE NANOOSE RANGE
*     DVP,EXT: DVP IDENTIFIER (FROM CONTROL PROGRAM) AND EXTRAPOLATION
*             CONTROL
*     ORIGIN: TRUE TARGET POSITION, USED FOR COMPARATIVE ANALYSIS
*  OUTPUTS:
*     THER  : ELEVATION ANGLE ERROR (LBL - SBL)
*     PHER  : AZIMUTH   ANGLE ERROR (LBL - SBL)
*     HER   : HORIZONTAL RANGE ERROR (LBL - SBL)
*     XER   : X COORDINATE ERROR (LBL - SBL)
*     YER   : Y COORDINATE ERROR (LBL - SBL)
*     ZER   : Z COORDINATE ERROR (LBL - SBL)
*    UPDATED VALUES FOR XS,XL,PHS,PHL,THS,THL,HS,HL
*-----------------------------------------------------------------------
*                                                 J.A. GEMBARSKI
*                                                 NPS   01/01/92
*-----------------------------------------------------------------------

      DIMENSION CPH(7,3),THER(6),PHER(6),HER(6),XER(6),
     &          YER(6),ZER(6),A(3),X(3),XS(18,3),XL(6,3),
     &          THS(18),THL(18),PHS(18),PHL(18),HS(18),HL(18),
     &          NEW(3),ORIGIN(6,3),SHIFT(7,6),ERROR(7,6),SENSR(7),
     &          XPH(7,3),YPH(7,3),ZPH(7,3),TIME(18,4),J(6),LINKS(18)


      DOUBLE PRECISION    CPH,THER,PHER,HER,XER,YER,ZER,A,X,XS
      DOUBLE PRECISION    XPH,YPH,ZPH
      DOUBLE PRECISION    XL,THS,THL,ORIGIN,SHIFT,ERROR
      DOUBLE PRECISION    PHS,PHL,HS,HL,TIME,NEW
      INTEGER   K,J,LINKS,SENSR,NR,EXT,DVP,C


      CALL UPDATE(SENSR,DVP,EXT,SHIFT,ERROR,CPH,XPH,YPH,ZPH,X,A)
      K =1
      DO 10 I=1,3*NR,3

         CALL SHIFTER(X(3),A(1),A(2),A(3),TIME,I,NEW,HS(I),THS(I))

         XS(I,1) = CPH(LINKS(I),1) + NEW(1)
         XS(I,2) = CPH(LINKS(I),2) + NEW(2)
         XS(I,3) = NEW(3)
C        WRITE(3,*) 'LONG FROM NEWERR'
```

```
         CALL LONG(TIME,LINKS,SENSR,CPH,K,XS,HS,THS,THL,PHS,PHL,
     &              HL,XL)

            THER(K) = THL(I) - THS(I)
            PHER(K) = PHL(I) - PHS(I)
            HER(K) =  HL(I) -  HS(I)
            XER(K) =  XL(K,1) -  XS(I,1)
            YER(K) =  XL(K,2) -  XS(I,2)
            ZER(K) =  XL(K,3) -  XS(I,3)
C     IF(C.GT.16)THEN
C        WRITE(91,*)
C        WRITE(91,230) J(K)
C        N = (3*K)-2
C      WRITE(91,231) (SENSR(LINKS(MP)),MP=N,N+2)
C      WRITE(91,232) XS(N,1),XS(N+1,1),XS(N+2,1),XL(K,1),ORIGIN(K,1)
C      WRITE(91,233) XS(N,2),XS(N+1,2),XS(N+2,2),XL(K,2),ORIGIN(K,2)
C      WRITE(91,234) XS(N,3),XS(N+1,3),XS(N+2,3),XL(K,3),ORIGIN(K,3)
C      WRITE(91,235) THS(N),THS(N+1),THS(N+2)
C      WRITE(91,236) THL(N),THL(N+1),THL(N+2)
C      WRITE(91,237) PHS(N),PHS(N+1),PHS(N+2)
C      WRITE(91,238) PHL(N),PHL(N+1),PHL(N+2)
C      WRITE(91,239) HS(N),HS(N+1),HS(N+2)
C      WRITE(91,240) HL(N),HL(N+1),HL(N+2)
C     ENDIF
         K = K + 1
   10 CONTINUE
C     IF(C.GT.14) THEN
C       WRITE(91,*)
C       WRITE(91,250)
C       WRITE(91,251)  (J(MP),MP=1,NR)
C       WRITE(91,*)
C       WRITE(91,252)  (THER(MP),MP=1,NR)
C       WRITE(91,*)
C       WRITE(91,253)  (PHER(MP),MP=1,NR)
C       WRITE(91,*)
C       WRITE(91,254)  (HER(MP),MP=1,NR)
C       WRITE(91,*)
C       WRITE(91,255)  (XER(MP),MP=1,NR)
C       WRITE(91,*)
C       WRITE(91,256)  (YER(MP),MP=1,NR)
C       WRITE(91,*)
C       WRITE(91,257)  (ZER(MP),MP=1,NR)
C     ENDIF
      RETURN
  230 FORMAT(25X,'REGION  ',I2)
  231 FORMAT(2X,3(3X,'ARRAY',1X,I2,1X,'SBL  '),3X,'LONGBASE',6X,'GOAL')
  232 FORMAT(1X,'X',4X,F9.2,2(7X,F9.2),5X,F9.2,3X,F9.2)
  233 FORMAT(1X,'Y',4X,F9.2,2(7X,F9.2),5X,F9.2,3X,F9.2)
  234 FORMAT(1X,'Z',4X,F9.2,2(7X,F9.2),5X,F9.2,3X,F9.2)
  235 FORMAT(1X,'THS',3X,F8.5,2(8X,F8.5),5X,F8.4,3X,F8.4)
  236 FORMAT(1X,'THL',3X,F8.5,2(8X,F8.5),5X,F8.4,3X,F8.4)
  237 FORMAT(1X,'PHS',3X,F8.5,2(8X,F8.5),5X,F8.4,3X,F8.4)
  238 FORMAT(1X,'PHL',3X,F8.5,2(8X,F8.5),5X,F8.4,3X,F8.4)
  239 FORMAT(1X,'HS',4X,F8.2,2(8X,F8.2),5X,F8.2,3X,F8.2)
  240 FORMAT(1X,'HL',4X,F8.2,2(8X,F8.2),5X,F8.2,3X,F8.2)
  250 FORMAT(15X,'CENTER ARRAY ERROR ANALYSIS')
  251 FORMAT(1X,'REGION',5X,I2,5(8X,I2))
  252 FORMAT(1X,'THER',2X,6(2X,F8.5))
  253 FORMAT(1X,'PHER',2X,6(2X,F8.5))
  254 FORMAT(1X,'HER',3X,6(2X,F8.4))
  255 FORMAT(1X,'XER',2X,6(2X,F8.3))
```

```
256 FORMAT(1X,'YER',2X,6(2X,F8.3))
257 FORMAT(1X,'ZER',2X,6(2X,F8.3))
    END
```

APPENDIX D: COMPUTER PROGRAM CONTROL

This is the main driving program of the error correction methodology.

Additional control lines have been added than are presented in the logic flow chart (Figure 11). These control lines were needed during multiple sensor array correction attempts. The nature of the control structure is to determine the entry into ANNEAL. The entry requirements into ANNEAL have also been changed (from objective function value 0.5 to 1.0). This enabled faster solution convergence by preceding to ANNEAL quicker.

```
      PROGRAM CONTROL
*
*  THIS PROGRAM CONTROLS THE ERROR CORRECTION SCHEME FOR THE NANOOSE
*  RANGE.
*  A SIMULATION START (SIMSTRT) SUBROUTINE GENERATES TRANSIT TIMES FOR
*  EACH SENSOR IN THE RANGE TO ITS TRIPLE OVERLAP REGIONS R(ANGE)TIME
*  (THIS INFORMATION IS NORMALLY PROVIDED BY THE RANGE).
*  SENSOR OPERATIONAL POSITION (ERROR INDUCED FROM THE ACTUAL POSITION)
*  IS CREATED THROUGH THE RANGE ERROR (RERROR) SUBROUTINE.  SENSOR
*  ACTUAL POSITION IS FOUND IN THE SUBROUTINE ARRAY.
*                                        J.A. GEMBARSKI
*                                        NPS   03/01/92
*----------------------------------------------------------------------
      DIMENSION HOLD(6),XPHLOC(7,3),YPHLOC(7,3),ZPHLOC(7,3),CPHLOC(7,3),
     &          ORIGIN(6,3),XS(18,3),XL(6,3),RLINKS(0:57,18),J(6),
     &          L(300),VEL(300),VO(300),V1(300),LM(300),RORIGIN(27,3),
     &          DZ(300),SHIFT(7,6),RSHIFT(0:57,6),YER(6),ZER(6),
     &          THL(18),HS(18),THER(6),PHER(6),HER(6),XER(6),SENSR(7),
     &          THS(18),HL(18),PHS(18),PHL(18),ERROR(7,6),LINKS(18),
     &          RTIME(0:57,27,4),RSENSR(0:57,7),TIME(18,4),CHECKA(0:57),
     &          RJ(0:57,6),RNR(0:57),RDATA(0:57,27,6),ROBJ(0:57),W1(7),
     &          RERRTAB(0:57,27,6),RXS(0:57,27,3),RXL(27,3),MARK(0:57)
*
      COMMON /SET1/ L,VEL,VO,V1,DZ,LM,M

      INTEGER   SENSR,J,ERR,K,NR,H,N,M,LINKS,RJ,RLINKS,RNR,RSENSR,S,
     +          MP,EXT,I,DVP,SS,NRS,MARK,MA
*
      DOUBLE PRECISION    ORIGIN,XS,XL,XPHLOC,YPHLOC,ZPHLOC,CPHLOC,XER
      DOUBLE PRECISION    VO,V1,LM,L,VEL,DZ,RDATA,RERRTAB,THER,YER,ZER
      DOUBLE PRECISION    THL,HOLD,PHER,HER,RORIGIN,RSHIFT,RTIME,ROBJ
      DOUBLE PRECISION    TIME,PHS,PHL,HL,HS,THS,SHIFT,ERROR,RXS,RXL,W
      DOUBLE PRECISION    MAX,TOL,WMIN,W1
*
      LOGICAL CHECKA,CHECKO,MKANN
*--------------------------------------------------------------------
C                            INITIALIZATION
```

```
*--------------------------------------------------------------------
      DO 3 I=1,300
         L(I)   = 0.0
         VEL(I) = 0.0
         VO(I)  = 0.0
         V1(I)  = 0.0
         DZ(I)  = 0.0
         LM(I)  = 0.0
         IF(I.LE.58) CHECKA(I-1) = .FALSE.
    3 CONTINUE
      DATA THS/18*0/
      DATA THL/18*0/
      DATA PHS/18*0/
      DATA PHL/18*0/
      DATA HS/18*0/
      DATA HL/18*0/
      DATA XS/54*0/
      DATA XL/18*0/
      DATA SHIFT/42*0/
      DATA XPHLOC/21*0/
      DATA YPHLOC/21*0/
      DATA ZPHLOC/21*0/
      DATA CPHLOC/21*0/
      DATA RSHIFT/348*0/
      DATA RTIME/6264*0/
      DATA RDATA/9396*0/
      DATA RSENSR/406*-1/
      DATA RORIGIN/81*0/
      DATA MARK/58*0/
      ERR = 0
      TOL = 1.0D-4
      MA = 58
*
C  CHOSE THE DVP TO BE USED 0     ISOSPEED = 10
C                           0.02  ISOGRADIENT = 11
C                           0.04  ISOGRADIENT = 12
C                           0.06  ISOGRADIENT = 13
C                           0.08  ISOGRADIENT = 14
C                           0.083 ISOGRADIENT = 15
C                        REAL WATER COLUMN =  5
      DVP = 10

C  TO CANCEL THE DVP EXTRAPOLATION SET EXT TO 2
      EXT = 2
*
C     OPEN(UNIT=3,FILE='/LONG OUT A')
C     OPEN(UNIT=44,FILE='/REC METHOD A')
C     OPEN(UNIT=90,FILE='/INFO TSTR A')
      OPEN(UNIT=91,FILE='/DATA TSTR T')
      OPEN(UNIT=92,FILE='/VALID BIG T')

  666 CALL SIMSTRT(DVP,EXT,RTIME,RORIGIN)
      CALL SETUP(DVP,EXT,RTIME,RSHIFT,RSENSR,RLINKS,RJ,RDATA,
     &           RERRTAB,RXS,RXL,RNR,ROBJ)
      MKANN = .FALSE.
  999 MAX = 0.0D0
      CHECKO = .TRUE.
      DO 40 I = 0,57
         IF(ROBJ(I).EQ.-1.0) GOTO 40
         IF(ROBJ(I).GT.5.0D0) PRINT*,'CHECK FOR',I
         IF(ROBJ(I).GT.1.0 .AND. .NOT.MKANN .AND. RNR(I).EQ.6) THEN
```

```
           PRINT*,'OBJ PREVENTIN ANNEAL',I,ROBJ(I)
           CHECKO = .FALSE.
         ENDIF
         IF(MA.EQ.I) GOTO 40
         IF(ROBJ(I).GE.MAX .AND. ROBJ(I).GT.TOL .AND. (RNR(I).EQ.6))THEN
           S = I
           MAX = ROBJ(I)
         ENDIF
   40 CONTINUE
      PRINT*,'S IN CONTROL',S
*
C     RETRIEVE RANGE INFORMATION ARRAYS FROM SETUP SUBROUTINE
*
            PRINT*,'BEFORE ERRFIX/ANNEAL'
         DO 5 I = 1,7
            SENSR(I) = RSENSR(S,I)
            PRINT*,'OBJ',SENSR(I),ROBJ(SENSR(I))
    5    CONTINUE

         DO 4 I =1,18
            LINKS(I) = RLINKS(S,I)
    4    CONTINUE

*
C    ESTABLISH INDUCED ERRORS FOR THE CENTER AND SATELLITE ARRAYS
*
         DO 9 I=1,7
            IF(SENSR(I).EQ.-1) GOTO 9
            CALL RERROR(SENSR(I),HOLD,ERR)
            DO 8 H = 1,6
               ERROR(I,H) = HOLD(H)
    8       CONTINUE
    9    CONTINUE
*
C      RETREIVE THE ARRAY SHIFT INFORMATION FOR PREVIOUS SHIFTS
*
         DO 7 I=1,7
            IF(SENSR(I).EQ.-1) GOTO 7
            DO 6 H = 1,6
               SHIFT(I,H) = RSHIFT(SENSR(I),H)
               IF(I.EQ.1) J(H) = RJ(S,H)
    6       CONTINUE
    7    CONTINUE

         K = 1
         DO 10 I = 1,18
            DO 11 H = 1,4
               TIME(I,H) = RTIME(SENSR(LINKS(I)),J(K),H)
   11       CONTINUE
            IF(MOD(I,3).EQ.0)K = K+1
   10    CONTINUE

         NR = RNR(S)


         DO 13 I = 1,6
            DO 14 H = 1,3
               ORIGIN(I,H) = RORIGIN(J(I),H)
   14       CONTINUE
   13    CONTINUE
*
```

```
C         RETREIVE THE WORKING ARRAYS FROM THE RANGE ARRAYS
*

           K = 1
        DO 20 I = 1,18
           THS(I) = RDATA(SENSR(LINKS(I)),J(K),1)
           THL(I) = RDATA(SENSR(LINKS(I)),J(K),2)
           PHS(I) = RDATA(SENSR(LINKS(I)),J(K),3)
           PHL(I) = RDATA(SENSR(LINKS(I)),J(K),4)
            HS(I) = RDATA(SENSR(LINKS(I)),J(K),5)
            HL(I) = RDATA(SENSR(LINKS(I)),J(K),6)
          XS(I,1) = RXS(SENSR(LINKS(I)),J(K),1)
          XS(I,2) = RXS(SENSR(LINKS(I)),J(K),2)
          XS(I,3) = RXS(SENSR(LINKS(I)),J(K),3)
           IF(MOD(I,3).EQ.0) K = K+1
   20   CONTINUE

        DO 21 I = 1,6
          THER(I) = RERRTAB(S,J(I),1)
          PHER(I) = RERRTAB(S,J(I),2)
           HER(I) = RERRTAB(S,J(I),3)
           XER(I) = RERRTAB(S,J(I),4)
           YER(I) = RERRTAB(S,J(I),5)
           ZER(I) = RERRTAB(S,J(I),6)
           XL(I,1) = RXL(J(I),1)
           XL(I,2) = RXL(J(I),2)
           XL(I,3) = RXL(J(I),3)
   21   CONTINUE

        WRITE(92,220)
        WRITE(92,221) (SENSR(N),N=1,7)
        WRITE(92,222)
        WRITE(92,223) (ERROR(N,1),N=1,7)
        WRITE(92,224) (ERROR(N,2),N=1,7)
        WRITE(92,225) (ERROR(N,3),N=1,7)
        WRITE(92,226) (ERROR(N,4),N=1,7)
        WRITE(92,227) (ERROR(N,5),N=1,7)
        WRITE(92,228) (ERROR(N,6),N=1,7)

        DO 77 K = 1,NR

         WRITE(92,*)
         WRITE(92,230) J(K)
         N = (3*K)-2
         WRITE(92,231) (SENSR(LINKS(MP)),MP=N,N+2)
         WRITE(92,232) XS(N,1),XS(N+1,1),XS(N+2,1),XL(K,1),ORIGIN(K,1)
         WRITE(92,233) XS(N,2),XS(N+1,2),XS(N+2,2),XL(K,2),ORIGIN(K,2)
         WRITE(92,234) XS(N,3),XS(N+1,3),XS(N+2,3),XL(K,3),ORIGIN(K,3)
         WRITE(92,235) THS(N),THS(N+1),THS(N+2)
         WRITE(92,236) THL(N),THL(N+1),THL(N+2)
         WRITE(92,237) PHS(N),PHS(N+1),PHS(N+2)
         WRITE(92,238) PHL(N),PHL(N+1),PHL(N+2)
         WRITE(92,239) HS(N),HS(N+1),HS(N+2)
         WRITE(92,240) HL(N),HL(N+1),HL(N+2)
   77  CONTINUE


      WRITE(92,*)
      WRITE(92,*)
      WRITE(92,250)
      WRITE(92,251) (J(MP),MP=1,NR)
      WRITE(92,*)
      WRITE(92,252) ((THL(3*MP-2)-THS(3*MP-2)),MP=1,NR)
      WRITE(92,*)
      WRITE(92,253) ((PHL(3*MP-2)-PHS(3*MP-2)),MP=1,NR)
      WRITE(92,*)
      WRITE(92,254) ((HL(3*MP-2)-HS(3*MP-2)),MP=1,NR)
      WRITE(92,*)
      WRITE(92,255) ((XL(MP,1)-XS((3*MP)-2,1)),MP=1,NR)
      WRITE(92,*)
      WRITE(92,256) ((XL(MP,2)-XS((3*MP)-2,2)),MP=1,NR)
      WRITE(92,*)
      WRITE(92,257) ((XL(MP,3)-XS((3*MP)-2,3)),MP=1,NR)

      DO 19 I=1,6
         HOLD(I) = SHIFT(1,I)
   19 CONTINUE
*----------------------------------------------------------------------
C                       ERROR CORRECTIONS
*----------------------------------------------------------------------
      IF(.NOT. CHECKO) THEN
        CALL ERRFIX(TIME,J,LINKS,SENSR,DVP,EXT,NR,ORIGIN,XS,XL,THS,THL,
     &              PHS,PHL,HS,HL,SHIFT,ERROR)
      ENDIF

      CALL EVAL(RDATA,RERRTAB,SENSR,LINKS,RNR,RJ,PHS,PHL,HL,HS,
     &                XS,XL,W1)

      IF(CHECKO) THEN
        PRINT*,'PRE-ANNEAL MARK',S,MARK(S)
        WMIN = 0.0D0
        DO 22 I = 2,7
           IF(SENSR(I).EQ.-1 .AND. RNR(I).NE.6) GOTO 22
           IF(ROBJ(SENSR(I)).GT.WMIN) WMIN = ROBJ(SENSR(I))
   22   CONTINUE
        WMIN = WMIN/2.0
        PRINT*,'TO ANNEAL',S,'WMIN',WMIN
        CALL ANNEAL(TIME,J,LINKS,SENSR,DVP,EXT,NR,ORIGIN,XS,XL,THS,THL,
     &              PHS,PHL,HS,HL,SHIFT,ERROR,WMIN)
        MKANN = .TRUE.
        MA = S
   23 ENDIF
*----------------------------------------------------------------------
C                OUTPUT THE NEW (CORRECTED) POSLOC DATA
*----------------------------------------------------------------------
        DO 78 K = 1,NR

         WRITE(92,*)
         WRITE(92,230) J(K)
         N = (3*K)-2
         WRITE(92,231) (SENSR(LINKS(MP)),MP=N,N+2)
         WRITE(92,232) XS(N,1),XS(N+1,1),XS(N+2,1),XL(K,1),ORIGIN(K,1)
         WRITE(92,233) XS(N,2),XS(N+1,2),XS(N+2,2),XL(K,2),ORIGIN(K,2)
         WRITE(92,234) XS(N,3),XS(N+1,3),XS(N+2,3),XL(K,3),ORIGIN(K,3)
         WRITE(92,235) THS(N),THS(N+1),THS(N+2)
         WRITE(92,236) THL(N),THL(N+1),THL(N+2)
         WRITE(92,237) PHS(N),PHS(N+1),PHS(N+2)
         WRITE(92,238) PHL(N),PHL(N+1),PHL(N+2)
         WRITE(92,239) HS(N),HS(N+1),HS(N+2)
         WRITE(92,240) HL(N),HL(N+1),HL(N+2)
   78  CONTINUE
```

```
      WRITE(92,*)
      WRITE(92,*)
      WRITE(92,250)
      WRITE(92,251) (J(MP),MP=1,NR)
      WRITE(92,*)
      WRITE(92,252) ((THL(3*MP-2)-THS(3*MP-2)),MP=1,NR)
      WRITE(92,*)
      WRITE(92,253) ((PHL(3*MP-2)-PHS(3*MP-2)),MP=1,NR)
      WRITE(92,*)
      WRITE(92,254) ((HL(3*MP-2)-HS(3*MP-2)),MP=1,NR)
      WRITE(92,*)
      WRITE(92,255) ((XL(MP,1)-XS((3*MP)-2,1)),MP=1,NR)
      WRITE(92,*)
      WRITE(92,256) ((XL(MP,2)-XS((3*MP)-2,2)),MP=1,NR)
      WRITE(92,*)
      WRITE(92,257) ((XL(MP,3)-XS((3*MP)-2,3)),MP=1,NR)

        DO 33 I =1,7
          IF(SENSR(I).EQ.-1) GOTO 33
          DO 34 H =1,6
            RSHIFT(SENSR(I),H) = SHIFT(I,H)
   34     CONTINUE
   33   CONTINUE
*
         K = 1
         DO 60 I = 1,18
            RDATA(SENSR(LINKS(I)),J(K),1) = THS(I)
            RDATA(SENSR(LINKS(I)),J(K),2) = THL(I)
            RDATA(SENSR(LINKS(I)),J(K),3) = PHS(I)
            RDATA(SENSR(LINKS(I)),J(K),4) = PHL(I)
            RDATA(SENSR(LINKS(I)),J(K),5) =  HS(I)
            RDATA(SENSR(LINKS(I)),J(K),6) =  HL(I)
            RXS(SENSR(LINKS(I)),J(K),1)   =  XS(I,1)
            RXS(SENSR(LINKS(I)),J(K),2)   =  XS(I,2)
            RXS(SENSR(LINKS(I)),J(K),3)   =  XS(I,3)
            RERRTAB(SENSR(LINKS(I)),J(K),1) = THL(I) - THS(I)
            RERRTAB(SENSR(LINKS(I)),J(K),2) = PHL(I) - PHS(I)
            RERRTAB(SENSR(LINKS(I)),J(K),3) =  HL(I) -  HS(I)

            RERRTAB(SENSR(LINKS(I)),J(K),4) =  XL(K,1)   -  XS(I,1)
            RERRTAB(SENSR(LINKS(I)),J(K),5) =  XL(K,2)   -  XS(I,2)
            RERRTAB(SENSR(LINKS(I)),J(K),6) =  XL(K,3)   -  XS(I,3)
            IF(MOD(I,3).EQ.0) K = K+1
   60    CONTINUE

         DO 61 K = 1,6
            RXL(J(K),1)     =  XL(K,1)
            RXL(J(K),2)     =  XL(K,2)
            RXL(J(K),3)     =  XL(K,3)
   61    CONTINUE

      MARK(S) =  MARK(S) + 1
         DO 41 K = 1,7
           IF(SENSR(K).EQ.-1) GOTO 41
           SS = SENSR(K)
           W = 0.0D0
           DO 42 I=1,RNR(SS)
              W = W + DSQRT( (RERRTAB(SS,RJ(SS,I),6))**2 +
     &                       (RERRTAB(SS,RJ(SS,I),3))**2 +
     &             (RDATA(SS,RJ(SS,I),5)*RERRTAB(SS,RJ(SS,I),2))**2)
   42      CONTINUE
```

```
           ROBJ(SS) = W/REAL(RNR(SS))
           PRINT*,'OBJ',SS,ROBJ(SS)
   41    CONTINUE
      PRINT*,'MARK ',S,MARK(S)

      IF(MARK(S).LT.10) GOTO 999

      STOP
*
  220 FORMAT(11X,'CENTER',20X,'ADJACENTS')
  221 FORMAT(1X,'ARRAYS',6X,I2,9X,I2,5(6X,I2))
  222 FORMAT(/,1X,'SHIFTS')
  223 FORMAT(3X,'DELX',4X,F8.4,2X,6(1X,F8.4))
  224 FORMAT(3X,'DELY',4X,F8.4,2X,6(1X,F8.4))
  225 FORMAT(3X,'DELZ',4X,F8.4,2X,6(1X,F8.4))
  226 FORMAT(3X,'DXTILT',2X,F8.4,2X,6(1X,F8.4))
  227 FORMAT(3X,'DYTILT',2X,F8.4,2X,6(1X,F8.4))
  228 FORMAT(3X,'DZROT',3X,F8.4,2X,6(1X,F8.4))
  230 FORMAT(25X,'REGION  ',I2)
  231 FORMAT(2X,3(3X,'ARRAY',1X,I2,1X,'SBL '),3X,'LONGBASE',6X,'GOAL')
  232 FORMAT(1X,'X',4X,F9.2,2(7X,F9.2),5X,F9.2,3X,F9.2)
  233 FORMAT(1X,'Y',4X,F9.2,2(7X,F9.2),5X,F9.2,3X,F9.2)
  234 FORMAT(1X,'Z',4X,F9.2,2(7X,F9.2),5X,F9.2,3X,F9.2)
  235 FORMAT(1X,'THS',3X,F8.5,2(8X,F8.5),5X,F8.4,3X,F8.4)
  236 FORMAT(1X,'THL',3X,F8.5,2(8X,F8.5),5X,F8.4,3X,F8.4)
  237 FORMAT(1X,'PHS',3X,F8.5,2(8X,F8.5),5X,F8.4,3X,F8.4)
  238 FORMAT(1X,'PHL',3X,F8.5,2(8X,F8.5),5X,F8.4,3X,F8.4)
  239 FORMAT(1X,'HS',4X,F8.2,2(8X,F8.2),5X,F8.2,3X,F8.2)
  240 FORMAT(1X,'HL',4X,F8.2,2(8X,F8.2),5X,F8.2,3X,F8.2)
  250 FORMAT(15X,'CENTER ARRAY ERROR ANALYSIS')
  251 FORMAT(1X,'REGION',5X,I2,5(8X,I2))
  252 FORMAT(1X,'THER',2X,6(2X,F8.5))
  253 FORMAT(1X,'PHER',2X,6(2X,F8.5))
  254 FORMAT(1X,'HER',3X,6(2X,F8.4))
  255 FORMAT(1X,'XER',2X,6(2X,F8.3))
  256 FORMAT(1X,'YER',2X,6(2X,F8.3))
  257 FORMAT(1X,'ZER',2X,6(2X,F8.3))

      END
```

APPENDIX E: COMPUTER PROGRAMS - GENERAL RANGE AND SYSTEM SUPPORTING

This appendix contains the following FORTRAN programs:

UPDATE, FIND, PLANE, RAYFIT (formally RAYFIT1), ISOGAD (formally ISGRAD1), ARRAY, FINDOVR, OVERLAP, POINTST, GLOBE, RERROR.

```
    SUBROUTINE UPDATE(SENSR,DVP,EXT,SHIFT,ERROR,CPH,XPH,YPH,ZPH,
     &                  PU,AU)
*  THIS SUBROUTINE UPDATES THE POSITION OF THE HYDROPHONES FOR THE
*  CENTER ARRAY AND ALL OF ITS SATELLITE ARRAYS.  POSITIONS ARE
*  PROVIDED IN GLOBAL RANGE COORDINATES.
*                                         J.A. GEMBARSKI
*                                         NPS   01/10/92
*---------------------------------------------------------------------
*  INPUTS:
*     SENSR  : 7 ELEMENT ARRAY CONTAINING CENTER (1) AND SATELLITE
*              (2-7) SENSOR NUMBERS
*     EXT    : EXTRAPOLATION CONTROL FOR DVP PROFILE  1-EXTRAPOLATE
*              IF NEEDED, 2- DO NOT EXTRAPOLATE
*     SHIFT  : ARRAY CONTAINING SHIFTS TO BE PREFORMED ON CENTER AND
*              SATELLITE ARRAYS (FIRST ELEMENT) IN THE X,Y,Z (SECOND
*              ELEMENT, 1-3) AND THE XTILT,YTILT,ZROT (4-6)
*     ERROR  : ARRAY CONTAINING ERRORS INDUCED ON CENTER AND
*              SATELLITE ARRAYS (FIRST ELEMENT) IN THE X,Y,Z (SECOND
*              ELEMENT, 1-3) AND THE XTILT,YTILT,ZROT (4-6)
*  OUTPUTS:
*     CPH,XPH,YPH,ZPH : GLOBAL POSITIONS OF THE C,X,Y,Z HYDROPHONES FOR
*              CENTER AND SATELLITE SENSORS (FIRST ELEMENT 1-7)
*     PU,AU  : CURRENT DATA FOR THE CENTER SENSOR PU-POSITION, AU-
*              ANGULAR ORIENTATION
*---------------------------------------------------------------------
      DIMENSION CPH(7,3),SHIFT(7,6),X(3),A(3),ERROR(7,6),
     &          XPH(7,3),YPH(7,3),ZPH(7,3),PU(3),AU(3),
     &          CP(3),XP(3),YP(3),ZP(3),SENSR(7)


      DOUBLE PRECISION CPH,XPH,YPH,ZPH,X,A,AU,PU,SHIFT,
     &                 XP,YP,ZP,CP,ERROR

      INTEGER  SENSR,I,ERR,EXT,DVP
      CHARACTER*15 NAME1
*
C         UPDATE POSITIONS OF C PHONES FROM PREVIOUS SHIFTS
*
      I = 0
    5 IF(SENSR(I+1).NE.0)THEN
         I = I + 1
         CALL ARRAY(SENSR(I),X(1),X(2),X(3),A(1),A(2),A(3),ERR)
           X(1) = X(1) + ERROR(I,1)  + SHIFT(I,1)
           X(2) = X(2) + ERROR(I,2)  + SHIFT(I,2)
           X(3) = X(3) + ERROR(I,3)  + SHIFT(I,3)
           A(1) = A(1) + ERROR(I,4)  + SHIFT(I,4)
           A(2) = A(2) + ERROR(I,5)  + SHIFT(I,5)
```

```
          A(3) = A(3) + ERROR(I,6)  + SHIFT(I,6)
        IF(I.EQ.1) THEN
          DO 10 N =1,3
             PU(N) = X(N)
             AU(N) = A(N)
10        CONTINUE
        ENDIF
        CALL GLOBE(X,A,XP,YP,ZP,CP)
        DO 7 LP=1,3
           CPH(I,LP) = CP(LP)
           XPH(I,LP) = XP(LP)
           YPH(I,LP) = YP(LP)
           ZPH(I,LP) = ZP(LP)
 7      CONTINUE
      IF(I.LE.6)GOTO 5
     ENDIF
     IF(SHIFT(1,3).GT.0 .AND. EXT.NE.2) THEN
       EXT = 1
       CALL VELPRO(0,NAME1,DVP,EXT,PU(3))
     ENDIF

     RETURN
     END
```

```
***********************************************************************
      SUBROUTINE FIND(AR,K,MI,MA)
*  FIND THE MINIMUM AND MAXIMUM OF THE ARRAY AR OF SIZE K
*                                         J.A. GEMBARSKI
*                                         NPS   01/01/92
*----------------------------------------------------------------------
      DIMENSION AR(K)
      DOUBLE PRECISION  AR,MI,MA
      INTEGER K,I

      MI = AR(1)
      MA = AR(1)
      DO 10 I=2,K
         IF(AR(I).GT.MA) MA = AR(I)
         IF(AR(I).LT.MI) MI = AR(I)
   10 CONTINUE
      RETURN
      END
**********************************************************************
      SUBROUTINE PLANE(X,Y,CO,A,B,C,E)

*  THIS SUBROUTINE FINDS THE EQUATION OF A PLANE,
*  PASSING THROUGH THE THREE POINTS X,Y,CO
*                                            J.A. GEMBARSKI
*                                            NPS   01/01/92
*------------------------------------------------------------------
      DIMENSION X(3),Y(3),CO(3)
      DOUBLE PRECISION X,Y,CO,A,B,C,E

      A =   X(2) * (Y(3) - CO(3))
     &    - X(3) * (Y(2) - CO(2))
     &    + (Y(2) * CO(3) - Y(3) * CO(2))

      B = - X(1) * (Y(3) - CO(3))
     &    + X(3) * (Y(1) - CO(1))
     &    - (Y(1) * CO(3) - Y(3) * CO(1))

      C =   X(1) * (Y(2) - CO(2))
     &    - X(2) * (Y(1) - CO(1))
     &    + (Y(1) * CO(2) - Y(2) * CO(1))

      E = - X(1) * (Y(2) * CO(3) - Y(3) * CO(2))
     &    + X(2) * (Y(1) * CO(3) - Y(3) * CO(1))
     &    - X(3) * (Y(1) * CO(2) - Y(2) * CO(1))
      RETURN
      END
```

```
***********************************************************************
      SUBROUTINE RAYFIT1(A1,A2,P1,P2,M,VEL,LM,DZ,V0,V1,T0,TH0,
     +                   TH1,IEST)
C                                                      09/12/89
C   NEW SUBROUTINE TO REPLACE TGEN, RAYTRACING ALGORITHM.
C----------------------------------------------------------------------
C   NO CHANGES HAVE BEEN MADE TO THIS SUBROUTINE.  RECIEVED FROM
C   LIB12.FOR OF REF 1
C                                          J.A. GEMBARSKI
C                                          NPS   01/01/92
C *********************************************************************
C       INPUTS:
C           A1,A2 - POSITION OF SENSOR (A2 > 0 DOWN)
C           P1,P2 - POSITION OF SOUND SOURCE ( P2 > 0 DOWN )
C           LM    - ARRAY CONTAINING LAYER MIDPOINTS
C           M     - NUMBER OF LAYER MIDPOINTS
C           VEL   - ARRAY CONTAINING SOUND VELOCITY AT THE
C                     LAYER MIDPOINTS.
C           V0    - SPEED INTERCEPT VALUES
C           V1    - SPEED SLOPE VALUES
C           DZ    - DEPTH INCREMENTS
C           IEST  - FLAG FOR INITIALIZING THE ANGLE
C       OUTPUTS:
C           T0    - TRANSIT TIME
C           TH0   - ELEVATION ANGLE AT THE SENSOR
C           TH1   - ELEVATION ANGLE AT THE SOUND SOURCE
C ***************************************************************

      DOUBLE PRECISION VEL(300),DZ(300),LM(300),V0(300)
      DOUBLE PRECISION V1(300),ANG(300)
      DOUBLE PRECISION A1,A2,P1,P2,T0,TH0,TH1,EP,S,C
      DOUBLE PRECISION H,H0,DW,VA2,VP2,R,Z,RV,Q1,Q2
      INTEGER M,IS,I,IEST,J,N

      EP = 1D-6

C   DETERMINE LAYERS INVOLVED IN RAY FITTING
      N = M
      J = M
      DO 30 I=1,M - 1
         IF ((LM(I).LE.A2).AND.(LM(I+1).GT.A2)) N=I
         IF ((LM(I).LE.P2).AND.(LM(I+1).GT.P2)) J=I
   30 CONTINUE

C   MAKE END CORRECTIONS FOR THE LAYERS
      DZ(N) = A2 - LM(N)
      DZ(J) = LM(J+1) - P2

C       COMPUTE SPEED OF SOUND AT A2 AND P2
      VA2 = V0(N) + V1(N)*A2
      VP2 = V0(J) + V1(J)*P2
      IF(IEST.NE.0) GOTO 50

C   INITIALIZE THE ELEVATION ANGLE AT THE SENSOR, TH0, BY
C   FITTING A STRAIGHT LINE SPEED PROFILE BETWEEN P2 AND A2.
      IF(VEL(N).EQ.VEL(J)) THEN
         TH0 = DATAN((A2-P2)/(P1-A1))
      ELSE
         Q2 = (VEL(N)*LM(J) - VEL(J)*LM(N)) / (VEL(N)-VEL(J))
         Q1 = 0.5*(P1+A1)+(0.5*(P2-A2)*(P2+A2-2*Q2))/(P1-A1)
         TH0 = DATAN((Q1-A1)/(Q2-A2))
      ENDIF

C   OUTER LOOP: SET UP RAY FITTING FOR THO = ELEVATION ANGLE
   50 S = DSIN(THO)
      C = DSQRT(1.0 - S**2)
      I = N
      RV = C/VA2
      HO = A1
      Z = A2

   60 IF(V1(I).EQ.0.0) THEN
         DW = DZ(I)/S
         H = HO + DW*C
      ELSE
         Q2 = -VO(I)/V1(I)
         IF (Q2) 61,62,63
  61     IS = -1
         GOTO 64
  62     IS = 0
         GOTO 64
  63     IS = 1
  64     CONTINUE
         Q1 = HO + (Q2-Z)*S/C
         R = DSQRT((Q2-Z)**2 + (Q1-HO)**2)
         C = RV*VEL(I)
         S = DSQRT(1.0-C**2)
         H = Q1 - IS*R*S
      ENDIF

      IF (I.EQ.J) GOTO 80
      HO = H
      Z = LM(I)
      I = I - 1
      GOTO 60

   80 TH1 = DACOS(RV*VP2)

C   FRACTIONAL LAYER CORRECTION
      IF(V1(J).NE.0.0) H = Q1 - IS*R*DSIN(TH1)
      IF (ABS(H-P1).LT.EP) GOTO 90

C   RE-ESTIMATE THO
      THO = DATAN(DTAN(THO)*H/P1)
      GOTO 50

C   PREPARE FOR COMPUTATION OF TRANSIT TIME.
C   COLLECT EXIT AND ENTRANCE ANGLES.
   90 ANG(J) = TH1
      ANG(N+1) = THO
C     G(J) = 1/V1(J)
C     G(N+1) = -1.0/V1(N)
      DO 95 I = J+1,N
         ANG(I) = DACOS(RV*VEL(I))
C        G(I) = (1/V1(I)) - (1/V1(I-1))
   95 CONTINUE

C   COMPUTE TRANSIT TIME
      TO = 0.0D0
      DO 100 I = J,N
         IF(V1(I).EQ.0.0) THEN
           TO = TO + DZ(I)/(VO(I)*DSIN(ANG(I)))
```

```
            ELSE
              TO=TO + DLOG((DCOS(ANG(I+1))*(1.0+DSIN(ANG(I))))/
     *           ((1.0+DSIN(ANG(I+1)))*DCOS(ANG(I))))/V1(I)
            ENDIF
  100   CONTINUE

C     REMOVE THE END CORRECTIONS.
        DZ(J) = LM(J+1) - LM(J)
        DZ(N) = LM(N+1) - LM(N)
        IEST = 1

        RETURN
        END
```

```
******************************************************************
      SUBROUTINE ISOGAD1(A1,A2,TO,THO,N,LM,VEL,V0,V1,DZ,H,Z,TH1)
C-----------------------------------------------------------------
C   NO CHANGES HAVE BEEN MADE TO THIS SUBROUTINE.  RECIEVED FROM
C   LIB12.FOR OF REF 1 (ORIGINAL NAME: ISOGRAD1)
C                                          J.A. GEMBARSKI
C                                          NPS   01/01/92
C-----------------------------------------------------------------
C                                                        09/25/89
C     TO :    TRANSIT TIME (SEC).
C     THO : ELEVATION ANGLE AT SENSOR (RAD).
C     A1 :  HORIZONTAL COORDINATE OF SENSOR.
C     A2 :  VERTICAL COORDINATE OF SENSOR, POSITIVE DOWN.
C     V0,V1 : ARRAYS CONTAINING SOUND VELOCITY PARAMETERS.
C     LM :  ARRAY CONTAINING LAYER MIDPOINTS.
C     N :   INDEX OF DEEPEST LAYER USED.
C
C****************************************************************

      DIMENSION LM(300),V0(300),V1(300),DZ(300),VEL(300)
      DOUBLE PRECISION TO,H,HO,Z,A1,A2,THO,TH1,VEL
      DOUBLE PRECISION LM,V0,V1,DZ,Q1,Q2
      DOUBLE PRECISION VA2,R,TH,RV,DW,DT,X,T
      INTEGER N,IS,I

      I = N
      T = 0.0D0
      TH=THO
      HO = A1
      VA2=V0(I)+V1(I)*A2
      RV=DCOS(TH)/VA2
      Z = A2
      DZ(N) = Z - LM(N)
  50  IF(V1(I).EQ.0.0) THEN
         DW = DZ(I)/DSIN(TH)
         DT = DW/V0(I)
         H = HO + DW*DCOS(TH)
         TH1 = TH
      ELSE
         Q2=-V0(I)/V1(I)
         IF (Q2) 51,52,53
  51     IS = -1
         GOTO 54
  52     IS = 0
         GOTO 54
  53     IS = 1
  54     CONTINUE
         Q1=HO + (Q2-Z)*DTAN(TH)
         R=DSQRT((Q2-Z)**2 + (Q1-HO)**2)
         TH1=DACOS(RV*VEL(I))
         DT=DLOG((DCOS(TH)/(1.0+DSIN(TH)))*((1.0+DSIN(TH1))/
     *      DCOS(TH1)))/V1(I)
         H=Q1 - IS*R*DSIN(TH1)
      ENDIF
      T=T+DT
      IF (T.GE.TO) GOTO 60
      Z=LM(I)
      HO = H
      TH=TH1
      I=I-1
      GOTO 50
```

```
   60 DT=T0+DT-T
      IF(V1(I).EQ.0.0) THEN
         DW = V0(I)*DT
         DZ(I) = DW*DSIN(TH1)
         H = H0 + DW*DCOS(TH1)
         Z = Z - DZ(I)
      ELSE
         X=(EXP(DT*V1(I)))*(1+DSIN(TH))/DCOS(TH)
         TH1=DACOS((2*X)/(1+X**2))
         H = Q1 - IS*R*DSIN(TH1)
         Z = Q2 - IS*R*DCOS(TH1)
      ENDIF

C   RESTORE THE END LAYERS.

      DZ(I) = LM(I+1) - LM(I)
      DZ(N) = LM(N+1) - LM(N)

      RETURN
      END
```

```
************************************************************************
      SUBROUTINE ARRAY(AR,A1,A2,A3,XTILT,YTILT,ZROT,ERR)
C THIS SUBROUTINE PROVIDES RANGE COORDINATES OF THE ACOUSTIC
C CENTERS OF THE NANOOSE ARRAYS
C                                      J.A. GEMBARSKI
C                                      NPS   01/01/92
C----------------------------------------------------------------------

      REAL*8 A1,A2,A3,XTILT,YTILT,ZROT
      INTEGER*4 AR,ERR

C  ARRAY 0
      IF(AR.EQ.0) THEN
      A1 = 12188.01D0
      A2 = -131.52D0
      A3 = 1295.33D0
      XTILT =  0.002909
      YTILT =  0.014835
      ZROT  = -0.208183

C  ARRAY 1
      ELSEIF(AR.EQ.1) THEN
      A1 = 19463.16D0
      A2 = -174.99D0
      A3 = 1308.76D0
      XTILT =  0.061523
      YTILT = -0.036070
      ZROT  =  1.362579
C  ARRAY 2
      ELSEIF(AR.EQ.2) THEN
      A1 = 26991.39D0
      A2 = -109.83D0
      A3 = 1323.35D0
      XTILT =  0.000145
      YTILT =  0.005236
      ZROT  =  2.670336
C  ARRAY 3
      ELSEIF(AR.EQ.3) THEN
      A1 = 34505.10D0
      A2 = -80.76D0
      A3 = 1323.32D0
      XTILT =  0.027925
      YTILT = -0.011345
      ZROT  =  2.928139
C  ARRAY 4
      ELSEIF(AR.EQ.4) THEN
      A1 = 42005.19D0
      A2 = -55.17D0
      A3 = 1318.28D0
      XTILT =  0.001164
      YTILT = -0.040288
      ZROT  = -2.315877
C  ARRAY 5
      ELSEIF(AR.EQ.5) THEN
      A1 = 49497.00D0
      A2 = -25.23D0
      A3 = 1315.58D0
      XTILT = -0.000291
      YTILT = -0.004027
      ZROT  =  1.668535
C  ARRAY 6
```

```
      ELSEIF(AR.EQ.6) THEN
      A1 = 56972.28D0
      A2 = -21.21D0
      A3 = 1308.50D0
      XTILT =  0.013817
      YTILT =  0.041161
      ZROT  = -0.703420
C  ARRAY 7
      ELSEIF(AR.EQ.7) THEN
      A1 = 64680.66D0
      A2 =  15.33D0
      A3 = 1353.39D0
      XTILT =  0.034907
      YTILT =  0.022835
      ZROT  = -0.574144
C  ARRAY 8
      ELSEIF(AR.EQ.8) THEN
      A1 = 71969.73D0
      A2 = -29.28D0
      A3 = 1300.89D0
      XTILT = -0.005963
      YTILT = -0.012217
      ZROT  = -1.577341
C  ARRAY 9
      ELSEIF(AR.EQ.9) THEN
      A1 =     3.00D0
      A2 =     3.00D0
      A3 =     1.00D0
      XTILT =  0.000000
      YTILT =  0.000000
      ZROT  =  0.000000
C  ARRAY 10
      ELSEIF(AR.EQ.10) THEN
      A1 = 47100.00D0
      A2 = -3600.00D0
      A3 = 1300.00D0
      XTILT =  0.000000
      YTILT =  0.000000
      ZROT  =  0.000000
C  ARRAY 11
      ELSEIF(AR.EQ.11) THEN
      A1 = 23173.89D0
      A2 = -6488.40D0
      A3 = 1312.09D0
      XTILT = -0.004654
      YTILT =  0.000436
      ZROT  =  2.784376
C  ARRAY 12
      ELSEIF(AR.EQ.12) THEN
      A1 = 30731.25D0
      A2 = -6553.05D0
      A3 = 1312.90D0
      XTILT =  0.002036
      YTILT =  0.001745
      ZROT  = -3.042179
C  ARRAY 13
      ELSEIF(AR.EQ.13) THEN
      A1 = 38213.61D0
      A2 = -6640.77D0
      A3 = 1323.05D0
      XTILT =  0.000291
      YTILT =  0.006254
      ZROT  =  1.373522
C   ARRAY 14
      ELSEIF(AR.EQ.14) THEN
      A1 = 45647.07D0
      A2 = -6513.18D0
      A3 = 1324.78D0
      XTILT =  0.001309
      YTILT =  0.002327
      ZROT  = -2.348044
C   ARRAY 15
      ELSEIF(AR.EQ.15) THEN
      A1 = 53249.43D0
      A2 = -6354.60D0
      A3 = 1316.66D0
      XTILT =  0.003345
      YTILT =  0.004509
      ZROT  =  0.581544
C   ARRAY 16
      ELSEIF(AR.EQ.16) THEN
      A1 = 60859.74D0
      A2 = -6356.07D0
      A3 = 1313.42D0
      XTILT =  0.014835
      YTILT =  0.036943
      ZROT  =  2.303276
C   ARRAY 17
      ELSEIF(AR.EQ.17) THEN
      A1 = 68217.93D0
      A2 = -6524.10D0
      A3 = 1313.43D0
      XTILT =  0.008290
      YTILT =  0.034761
      ZROT  =  2.158449
C   ARRAY 54
      ELSEIF(AR.EQ.54) THEN
      A1 = 38029.95D0
      A2 =  5401.98D0
      A3 = 1212.69D0
      XTILT =  0.007709
      YTILT = -0.003782
      ZROT  = -1.056919
C   ARRAY 55
      ELSEIF(AR.EQ.55) THEN
      A1 = 45645.75D0
      A2 = 6369.66D0
      A3 = 1188.12D0
      XTILT =  0.027634
      YTILT =  0.039415
      ZROT  = -0.728553
C   ARRAY 56
      ELSEIF(AR.EQ.56) THEN
      A1 = 53180.13D0
      A2 =  6417.96D0
      A3 =  1218.84D0
      XTILT =  0.037525
      YTILT =  0.048142
      ZROT  = -1.392651
C   ARRAY 57
      ELSEIF(AR.EQ.57) THEN
      A1 = 60745.71D0
      A2 =  6419.40D0
      A3 = 1088.24D0
      XTILT =  0.006981
      YTILT =  0.001891
      ZROT  = -3.108606
C  ARRAY 23
      ELSEIF(AR.EQ.23) THEN
      A1 = 41605.14D0
      A2 =-12150.18D0
      A3 = 1268.23D0
      XTILT = -0.002182
      YTILT =  0.003200
      ZROT  = -1.845214
C  ARRAY 24
      ELSEIF(AR.EQ.24) THEN
      A1 = 49572.00D0
      A2 =-12966.00D0
      A3 = 1300.00D0
      XTILT = -0.007272
      YTILT =  0.055269
      ZROT  = -1.343904
C  ARRAY 25
      ELSEIF(AR.EQ.25) THEN
      A1 = 56993.79D0
      A2 =-12999.33D0
      A3 = 1205.48D0
      XTILT =  0.000291
      YTILT = -0.002182
      ZROT  = -0.593726
C  ARRAY 26
      ELSEIF(AR.EQ.26) THEN
      A1 = 64442.94D0
      A2 =-12971.04D0
      A3 = 1255.35D0
      XTILT = -0.014835
      YTILT = -0.012654
      ZROT  =  3.134192
C  ARRAY 27
      ELSEIF(AR.EQ.27) THEN
      A1 = 22119.60D0
      A2 =-15908.70D0
      A3 =  -83.00D0
      XTILT =  0.000000
      YTILT =  0.000000
      ZROT  =  0.000000
C  ARRAY 28
      ELSEIF(AR.EQ.28) THEN
      A1 = 45000.00D0
      A2 =  1500.00D0
      A3 = 1350.00D0
      XTILT =  0.000000
      YTILT =  0.000000
      ZROT  =  0.000000
C  ARRAY 29
      ELSEIF(AR.EQ.29) THEN
      A1 =     0.00D0
      A2 =     0.00D0
      A3 =    0.00D0
      XTILT =  0.000000
      YTILT =  0.000000
      ZROT  =  0.000000
```

```
ELSE
A1 = 0.0D0
A2 = 0.0D0
A3 = 0.0D0
XTILT =  0.000000
YTILT =  0.000000
ZROT  =  0.000000
ERR = 1
ENDIF
RETURN
END
```

```
***********************************************************************
      SUBROUTINE FINDOVR(I,REGION,ERR)
*
*       SUBROUTINE TO FIND ALL CROSSOVER REGIONS FOR THE GIVEN ARRAY
*     ON THE NANOOSE RANGE.
*                                          J.A. GEMBARSKI
*                                          NPS   01/10/92
*---------------------------------------------------------------------
      DIMENSION REGION(6)
      INTEGER*4 I,REGION,ERR,K
*
      DO 10 K =1,6
        REGION(K) = 0
   10 CONTINUE
      ERR  = 0
*
C     ARRAY  1
        IF(I .EQ. 1 )THEN
          REGION(1) = 1
C     ARRAY  2
        ELSEIF(I .EQ. 2 )THEN
          REGION(1) = 1
          REGION(2) = 2
          REGION(3) = 3
C     ARRAY  3
        ELSEIF(I .EQ. 3 )THEN
          REGION(1) = 3
          REGION(2) = 4
          REGION(3) = 5
          REGION(4) = 6
C     ARRAY  4
        ELSEIF(I .EQ. 4 )THEN
          REGION(1) = 5
          REGION(2) = 6
          REGION(3) = 7
          REGION(4) = 8
          REGION(5) = 9
          REGION(6) = 10
C     ARRAY  5
        ELSEIF(I .EQ. 5 )THEN
          REGION(1) = 8
          REGION(2) = 9
          REGION(3) = 14
          REGION(4) = 15
          REGION(5) = 16
          REGION(6) = 17
C     ARRAY  6
        ELSEIF(I .EQ. 6 )THEN
          REGION(1) = 15
          REGION(2) = 16
          REGION(3) = 18
          REGION(4) = 19
          REGION(5) = 20
          REGION(6) = 21
C     ARRAY  7
        ELSEIF(I .EQ. 7 )THEN
          REGION(1) = 19
          REGION(2) = 20
          REGION(3) = 26
          REGION(4) = 27
C     ARRAY  8
```

```
          ELSEIF(I .EQ. 8 )THEN
            REGION(1) = 27
C     ARRAY  11
          ELSEIF(I .EQ. 11)THEN
            REGION(1) = 1
            REGION(2) = 2
C     ARRAY  12
          ELSEIF(I .EQ. 12)THEN
            REGION(1) = 2
            REGION(2) = 3
            REGION(3) = 4
C     ARRAY  13
          ELSEIF(I .EQ. 13)THEN
            REGION(1) = 4
            REGION(2) = 5
            REGION(3) = 10
            REGION(4) = 11
C     ARRAY  14
          ELSEIF(I .EQ. 14)THEN
            REGION(1) = 9
            REGION(2) = 10
            REGION(3) = 11
            REGION(4) = 12
            REGION(5) = 13
            REGION(6) = 14
C     ARRAY  15
          ELSEIF(I .EQ. 15)THEN
            REGION(1) = 13
            REGION(2) = 14
            REGION(3) = 15
            REGION(4) = 21
            REGION(5) = 22
            REGION(6) = 23
C     ARRAY  16
          ELSEIF(I .EQ. 16)THEN
            REGION(1) = 20
            REGION(2) = 21
            REGION(3) = 22
            REGION(4) = 24
            REGION(5) = 25
            REGION(6) = 26
C     ARRAY  17
          ELSEIF(I .EQ. 17)THEN
            REGION(1) = 25
            REGION(2) = 26
            REGION(3) = 27
C     ARRAY  23
          ELSEIF(I .EQ. 23)THEN
            REGION(1) = 11
            REGION(2) = 12
C     ARRAY  24
          ELSEIF(I .EQ. 24)THEN
            REGION(1) = 12
            REGION(2) = 13
            REGION(3) = 23
C     ARRAY  25
          ELSEIF(I .EQ. 25)THEN
            REGION(1) = 22
            REGION(2) = 23
            REGION(3) = 24
C     ARRAY  26
```

```
        ELSEIF(I .EQ. 26)THEN
          REGION(1) = 24
          REGION(2) = 25
C     ARRAY  54
        ELSEIF(I .EQ. 54)THEN
          REGION(1) = 6
          REGION(2) = 7
C     ARRAY  55
        ELSEIF(I .EQ. 55)THEN
          REGION(1) = 7
          REGION(2) = 8
          REGION(3) = 17
C     ARRAY  56
        ELSEIF(I .EQ. 56)THEN
          REGION(1) = 16
          REGION(2) = 17
          REGION(3) = 18
C     ARRAY  57
        ELSEIF(I .EQ. 57)THEN
          REGION(1) = 18
          REGION(2) = 19
        ELSE
          ERR = 1
        ENDIF
      RETURN
      END
```

```
***********************************************************************
      SUBROUTINE OVERLAP(J,AR,ERR)
*
*       SUBROUTINE TO RETURN THE ARRAYS ASSOCIATED WITH THE INPUT
*     CROSSOVER REGION J FOR THE NANOOSE RANGE.
*                                                J.A. GEMBARSKI
*                                                NPS   01/01/92
*----------------------------------------------------------------------
      DIMENSION AR(3)
      INTEGER*4  ERR,J,AR
*
C     REGION 1
        IF(J .EQ. 1 )THEN
          AR(1) = 1
          AR(2) = 2
          AR(3) = 11
C     REGION 2
        ELSEIF(J .EQ. 2 )THEN
          AR(1) = 2
          AR(2) = 11
          AR(3) = 12
C     REGION 3
        ELSEIF(J .EQ. 3 )THEN
          AR(1) = 2
          AR(2) = 3
          AR(3) = 12
C     REGION 4
        ELSEIF(J .EQ. 4 )THEN
          AR(1) = 3
          AR(2) = 12
          AR(3) = 13
C     REGION 5
        ELSEIF(J .EQ. 5 )THEN
          AR(1) = 3
          AR(2) = 4
          AR(3) = 13
C     REGION 6
        ELSEIF(J .EQ. 6 )THEN
          AR(1) = 3
          AR(2) = 4
          AR(3) = 54
C     REGION 7
        ELSEIF(J .EQ. 7 )THEN
          AR(1) = 4
          AR(2) = 54
          AR(3) = 55
C     REGION 8
        ELSEIF(J .EQ. 8 )THEN
          AR(1) = 4
          AR(2) = 5
          AR(3) = 55
C     REGION 9
        ELSEIF(J .EQ. 9 )THEN
          AR(1) = 4
          AR(2) = 5
          AR(3) = 14
C     REGION 10
        ELSEIF(J .EQ. 10)THEN
          AR(1) = 4
          AR(2) = 13
          AR(3) = 14
```

```
C     REGION 11
        ELSEIF(J .EQ. 11)THEN
          AR(1) = 13
          AR(2) = 14
          AR(3) = 23
C     REGION 12
        ELSEIF(J .EQ. 12)THEN
          AR(1) = 14
          AR(2) = 23
          AR(3) = 24
C     REGION 13
        ELSEIF(J .EQ. 13)THEN
          AR(1) = 14
          AR(2) = 15
          AR(3) = 24
C     REGION 14
        ELSEIF(J .EQ. 14)THEN
          AR(1) = 5
          AR(2) = 14
          AR(3) = 15
C     REGION 15
        ELSEIF(J .EQ. 15)THEN
          AR(1) = 5
          AR(2) = 6
          AR(3) = 15
C     REGION 16
        ELSEIF(J .EQ. 16)THEN
          AR(1) = 5
          AR(2) = 6
          AR(3) = 56
C     REGION 17
        ELSEIF(J .EQ. 17)THEN
          AR(1) = 5
          AR(2) = 55
          AR(3) = 56
C     REGION 18
        ELSEIF(J .EQ. 18)THEN
          AR(1) = 6
          AR(2) = 56
          AR(3) = 57
C     REGION 19
        ELSEIF(J .EQ. 19)THEN
          AR(1) = 6
          AR(2) = 7
          AR(3) = 57
C     REGION 20
        ELSEIF(J .EQ. 20)THEN
          AR(1) = 6
          AR(2) = 7
          AR(3) = 16
C     REGION 21
        ELSEIF(J .EQ. 21)THEN
          AR(1) = 6
          AR(2) = 15
          AR(3) = 16
C     REGION 22
        ELSEIF(J .EQ. 22)THEN
          AR(1) = 15
          AR(2) = 16
          AR(3) = 25
C     REGION 23
```

```
        ELSEIF(J .EQ. 23)THEN
          AR(1) = 15
          AR(2) = 24
          AR(3) = 25
C     REGION 24
        ELSEIF(J .EQ. 24)THEN
          AR(1) = 16
          AR(2) = 25
          AR(3) = 26
C     REGION 25
        ELSEIF(J .EQ. 25)THEN
          AR(1) = 16
          AR(2) = 17
          AR(3) = 26
C     REGION 26
        ELSEIF(J .EQ. 26)THEN
          AR(1) = 7
          AR(2) = 16
          AR(3) = 17
C     REGION 27
        ELSEIF(J .EQ. 27)THEN
          AR(1) = 7
          AR(2) = 8
          AR(3) = 17
        ELSE
          AR(1) = 0
          AR(2) = 0
          AR(3) = 0
          ERR   = 1
        ENDIF
      RETURN
      END
```

```
************************************************************************
      SUBROUTINE POINTST(J,X0,Y0,Z0)
*
*     SUBROUTINE TO FIND THE INITIAL STARTING POINT (THE CENTER OF A
*     TRIPLE OVERLAP REGION) X0 AND Y0, FOR OVERLAP REGION J.
*     THE INITIAL STARTING POINT IS AT DEPTH X0 (400 FT) ARBITRARILY
*                                                J.A. GEMBARSKI
*                                                NPS   01/01/92
*-----------------------------------------------------------------------
      DIMENSION X(3),Y(3),Z(3),AR(3)
      REAL*8     X,Y,Z,X0,Y0,Z0,NUM1,NUM2,DEN,A,B,C
      INTEGER*4  J,AR,ERR
*
      CALL OVERLAP(J,AR,ERR)
*
      DO 10 I=1,3
          CALL ARRAY(AR(I),X(I),Y(I),Z(I),A,B,C,ERR)
   10 CONTINUE
*
C     SET THE DEPTH OF THE STARTING POINT
*
      Z0 = 400.0
*
      NUM1 = (X(2)**2-X(1)**2+Y(2)**2-Y(1)**2)/(X(2)-X(1))
      NUM2 = (X(3)**2-X(1)**2+Y(3)**2-Y(1)**2)/(X(3)-X(1))
      DEN  = 2*((Y(1)-Y(3))/(X(3)-X(1))-(Y(1)-Y(2))/(X(2)-X(1)))
*
      Y0 = (NUM1 - NUM2)/DEN
      X0 = NUM1/2 + Y0*(Y(1)-Y(2))/(X(2)-X(1))
*
      RETURN
      END
*************************************************************************
      SUBROUTINE GLOBE(P,A,XP,YP,ZP,CP)
*
*   THIS SUBROUTINE DETERMINES THE GLOBAL POSITIONS OF EACH HYDROPHONE
*   OF THE IMPUTED ARRAY.  INPUT POSITION IS THAT OF THE ACOUSTIC
*   CENTER.
*
* INPUTS
*    P: POSITION ARRAY P(1) P(2) P(3), X,Y,Z POSITIONS RESPECTIVELY
*    A: ORIENTATION ARRAY A(1),A(2),A(3) XTILT,YTILT,ZROT RESPECT.
*
* OUTPUTS
*   XP: POSITION OF THE X HYDROPHONE OF THE SENSOR  X,Y,Z COORDINATES
*       RESPECTIVELY
*   YP,ZP,CP: THE Y, Z, AND C HYDROPHONE POSITIONS
*                                              J.A. GEMBARSKI
*                                              NPS   01/01/92
*-----------------------------------------------------------------------
      DIMENSION B(5,3),XP(3),YP(3),ZP(3),CP(3),P(3),A(3)
      INTEGER I
      DOUBLE PRECISION P,A,B,S1,S2,S3,C1,C2,C3,D,XP,YP,ZP,CP
*
      D=30.0D0
*
*  FORM THE SINES AND COSINES OF ALL EULER ANGLES:ROLL,PITCH AND YAW
*
      S2=DSIN(A(1))
      C2=DSQRT(1-S2**2)
      S1=DSIN(A(2))/C2
      C1=DSQRT(1-S1**2)
      S3=-DSIN(A(3))
      C3=DCOS(A(3))
*
*                 FORM THE B TRANSITION MATRIX
*
      B(1,1)=C2*C3
      B(1,2)=C2*S3
      B(1,3)=S2
      B(2,1)=-S1*S2*C3-C1*S3
      B(2,2)=-S1*S2*S3+C1*C3
      B(2,3)=S1*C2
      B(3,1)=-C1*S2*C3+S1*S3
      B(3,2)=-C1*S2*S3-S1*C3
      B(3,3)=C1*C2
      DO 66 I=1,3
       B(4,I)=0.0D0
       B(5,I)= 0.5*(B(I,1) + B(I,2) + B(I,3))
   66 CONTINUE
C     PRINT*,'B1',B(1,1),B(1,2),B(1,3)
C     PRINT*,'B2',B(2,1),B(2,2),B(2,3)
C     PRINT*,'B3',B(3,1),B(3,2),B(3,3)
*
*             LOCATE THE X,Y,Z,C HYDROPHONE LOCATIONS
*
        XP(1)=P(1) + (D)*(B(1,1)-B(5,1))
        XP(2)=P(2) + (D)*(B(1,2)-B(5,2))
        XP(3)=P(3) + (D)*(B(5,3)-B(1,3))

        YP(1)=P(1) + (D)*(B(2,1)-B(5,1))
        YP(2)=P(2) + (D)*(B(2,2)-B(5,2))
        YP(3)=P(3) + (D)*(B(5,3)-B(2,3))

        ZP(1)=P(1) + (D)*(B(3,1)-B(5,1))
        ZP(2)=P(2) + (D)*(B(3,2)-B(5,2))
        ZP(3)=P(3) + (D)*(B(5,3)-B(3,3))

        CP(1)=P(1) + (D)*(B(4,1)-B(5,1))
        CP(2)=P(2) + (D)*(B(4,2)-B(5,2))
        CP(3)=P(3) + (D)*(B(5,3)-B(4,3))

      RETURN
      END
```

```
C==================================================================
      SUBROUTINE RERROR(AR,ERRAR,ERR)
C THIS SUBROUTINE PROVIDES RANGE INDUCED ERRORS TO SENSORS
C (DELX,DELY,DELZ,DXTILT,DYTILT,DZROT) FOR THE NANOOSE RANGE
C FOR INPUT INTO THE PROGRAM SIMSTRT
C                                J.A. GEMBARSKI
C                                NPS   01/01/92
C==================================================================
      DIMENSION ERRAR(6)
      DOUBLE PRECISION ERRAR,VAL
      INTEGER AR,ERR,NO
      CHARACTER*1 RES
C  ARRAY 0
      IF(AR.EQ.0) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 1
      ELSEIF(AR.EQ.1) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 2
      ELSEIF(AR.EQ.2) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 3
      ELSEIF(AR.EQ.3) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 4
      ELSEIF(AR.EQ.4) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 5
      ELSEIF(AR.EQ.5) THEN
C       ERRAR(1) = 7.9D0
C       ERRAR(2) =-6.2D0
C       ERRAR(3) =-1.8D0
C       ERRAR(4) = 0.007D0
C       ERRAR(5) =-0.003D0
C       ERRAR(6) = 0.001D0
```

```
          ERRAR(1) = 0.0D0
          ERRAR(2) = 0.0D0
          ERRAR(3) = 0.0D0
          ERRAR(4) = 0.0D0
          ERRAR(5) = 0.0D0
          ERRAR(6) = 0.0D0
C  ARRAY 6
        ELSEIF(AR.EQ.6) THEN
C         ERRAR(1) =-5.0D0
C         ERRAR(2) = 7.0D0
C         ERRAR(3) = 4.0D0
C         ERRAR(4) = 0.005D0
C         ERRAR(5) = 0.002D0
C         ERRAR(6) =-0.003D0
          ERRAR(1) = 0.0D0
          ERRAR(2) = 0.0D0
          ERRAR(3) = 0.0D0
          ERRAR(4) = 0.0D0
          ERRAR(5) = 0.0D0
          ERRAR(6) = 0.0D0
C  ARRAY 7
        ELSEIF(AR.EQ.7) THEN
          ERRAR(1) = 0.0D0
          ERRAR(2) = 0.0D0
          ERRAR(3) = 0.0D0
          ERRAR(4) = 0.0D0
          ERRAR(5) = 0.0D0
          ERRAR(6) = 0.0D0
C  ARRAY 8
        ELSEIF(AR.EQ.8) THEN
          ERRAR(1) = 0.0D0
          ERRAR(2) = 0.0D0
          ERRAR(3) = 0.0D0
          ERRAR(4) = 0.0D0
          ERRAR(5) = 0.0D0
          ERRAR(6) = 0.0D0
C  ARRAY 9
        ELSEIF(AR.EQ.9) THEN
          ERRAR(1) = 0.0D0
          ERRAR(2) = 0.0D0
          ERRAR(3) = 0.0D0
          ERRAR(4) = 0.0D0
          ERRAR(5) = 0.0D0
          ERRAR(6) = 0.0D0
C  ARRAY 10
        ELSEIF(AR.EQ.10) THEN
          ERRAR(1) = 0.0D0
          ERRAR(2) = 0.0D0
          ERRAR(3) = 0.0D0
          ERRAR(4) = 0.0D0
          ERRAR(5) = 0.0D0
          ERRAR(6) = 0.0D0
C  ARRAY 11
        ELSEIF(AR.EQ.11) THEN
          ERRAR(1) = 0.0D0
          ERRAR(2) = 0.0D0
          ERRAR(3) = 0.0D0
          ERRAR(4) = 0.0D0
          ERRAR(5) = 0.0D0
          ERRAR(6) = 0.0D0
C  ARRAY 12
```

```
      ELSEIF(AR.EQ.12) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 13
      ELSEIF(AR.EQ.13) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 14
      ELSEIF(AR.EQ.14) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C       ERRAR(1) = 4.0D0
C       ERRAR(2) =-8.0D0
C       ERRAR(3) = 5.0D0
C       ERRAR(4) =-0.009D0
C       ERRAR(5) =-0.004D0
C       ERRAR(6) = 0.003D0
C  ARRAY 15
      ELSEIF(AR.EQ.15) THEN
        ERRAR(1) = 11.0D0
        ERRAR(2) = 28.0D0
        ERRAR(3) = -2.0D0
        ERRAR(4) = 0.025D0
        ERRAR(5) = 0.010D0
        ERRAR(6) =-0.010D0
C       ERRAR(1) = 0.0D0
C       ERRAR(2) = 0.0D0
C       ERRAR(3) = 0.0D0
C       ERRAR(4) = 0.0D0
C       ERRAR(5) = 0.0D0
C       ERRAR(6) = 0.0D0
C  ARRAY 16
      ELSEIF(AR.EQ.16) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 17
      ELSEIF(AR.EQ.17) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 54
      ELSEIF(AR.EQ.54) THEN
```

```
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 55
      ELSEIF(AR.EQ.55) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 56
      ELSEIF(AR.EQ.56) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 57
      ELSEIF(AR.EQ.57) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 23
      ELSEIF(AR.EQ.23) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 24
      ELSEIF(AR.EQ.24) THEN
C       ERRAR(1) =-0.5D0
C       ERRAR(2) =-1.0D0
C       ERRAR(3) = 0.7D0
C       ERRAR(4) =-0.010D0
C       ERRAR(5) = 0.005D0
C       ERRAR(6) =-0.020D0
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 25
      ELSEIF(AR.EQ.25) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 26
```

```
      ELSEIF(AR.EQ.26) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 27
      ELSEIF(AR.EQ.27) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 28
      ELSEIF(AR.EQ.28) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
C  ARRAY 29
      ELSEIF(AR.EQ.29) THEN
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
      ELSE
        ERRAR(1) = 0.0D0
        ERRAR(2) = 0.0D0
        ERRAR(3) = 0.0D0
        ERRAR(4) = 0.0D0
        ERRAR(5) = 0.0D0
        ERRAR(6) = 0.0D0
      ERR = 1
      ENDIF
      RETURN
  100 FORMAT(A1)
      END
```

APPENDIX F: COMPUTER PROGRAMS SHIFTER AND LONG(BASE)

These FORTRAN programs are used to determin the SBL and LBL poslocs for the error correction methodology.

```
      SUBROUTINE SHIFTER(A2,XTILT,YTILT,ZROT,T,K,NEW,H2,THONE)
C****************************************************************
C
C  THIS PROGRAM IS A MODIFICATION OF ERRPROP (08/22/90) TO
C  SHIFT THE POSLOC WHEN THE ARRAY IS MOVED. THE ORIGIN IS OVER THE C-
C  HYDROPHONE AND THE PROPOSED SYSTEM IS APPLIED.  A2 IS THE DEPTH OF
C  THE GEOMETRIC CENTER OF THE ARRAY CUBE.  THE ACOUSTIC CENTER IS
C  THE C-HYDROPHONE.
C  TRANSIT TIME ERROR.
C   THIS IS FOR THE KTH POSLOC AS DEFINED FROM THE SOURCE PROGRAM
C   (NEWERR OR SETUP)
C                                           J.A. GEMBARSKI
C                                           NPS   01/01/92
C****************************************************************
C
C   INPUTS:
C     A2:    DEPTH OF THE CENTER OF THE ARRAY
C     XTILT,YTILT,ZROT:  ORIENTATION INFORMATION ABOUT THE
C           SENSING ARRAY (RADIANS)
C     D:     LENGTH OF ARRAY EDGES.
C     L:     DEPTH OF LAYER BOUNDARIES.
C     M:     NUMBER OF RECORDS IN THE VELOCITY DEPTH PROFILE.
C     VEL:   AVERAGE SPEED OF SOUND IN THE LAYERS.
C     K:     POSLOC TO BE SHIFTED
C     T:     RAY TRANSIT TIMES FROM EACH HYDROPHONE X,Y,Z,C
C
C   OUTPUTS:
C     NEW:   LOCAL COORDINATES OF THE SHIFTED POSLOC
C     H2:    THE HORIZONTAL RANGE TO THE SHIFTED POSLOC
C     THONE:THE 'TRUE' ELEVATION ANGLE TO THE SHIFTED POSLOC
C
C****************************************************************

      DIMENSION B(5,3),DZ(300),HD(5),L(300)
      DIMENSION LM(300),NEW(3)
      DIMENSION T(18,4)
      DIMENSION V0(300),V1(300),VEL(300),X0(3)

      COMMON /SET1/ L,VEL,V0,V1,DZ,LM,M


      DOUBLE PRECISION B,DZ,HD,L,LM,PH1
      DOUBLE PRECISION T,THONE,V0,V1
      DOUBLE PRECISION VEL,X0,NEW

      DOUBLE PRECISION A1,A2,A2M,C1,C2,C3,CAZ,CX,CX0,CY
      DOUBLE PRECISION CY0,CZ,CZ0,D
      DOUBLE PRECISION PIE,S1,S2,S3,SAZ
      DOUBLE PRECISION V,XTILT,YTILT,ZROT

      DOUBLE PRECISION H2,Z2,THE
      INTEGER K,J,M,N

      PIE = DACOS(-1.0D0)
      D = 30.0D0

C   FORM SINES AND COSINES OF ALL THE EULER ANGLES:ROLL,PITCH,YAW
      S2 = DSIN(XTILT)
      C2 = DSQRT(1.0 - S2**2)
      S1 = DSIN(YTILT)/C2
      C1 = DSQRT(1.0 - S1**2)
      S3 = -DSIN(ZROT)
      C3 = DCOS(ZROT)

C   IN THE COORDINATE SYSTEM HAVING CENTER AT THE C-HYDROPHONE
C   AND POSITIVE-UPWARD, THE LOCATIONS OF THE FOUR HYDROPHONES
C   (RELATIVE TO THE ARM LENGTH D) ARE DEVELOPED NEXT.
      B(1,1) = C2*C3
      B(1,2) = C2*S3
      B(1,3) = S2
      B(2,1) = -S1*S2*C3 - C1*S3
      B(2,2) = -S1*S2*S3 + C1*C3
      B(2,3) = S1*C2
      B(3,1) = -C1*S2*C3 + S1*S3
      B(3,2) = -C1*S2*S3 - S1*C3
      B(3,3) = C1*C2

C   LIKE NOTATION WILL BE USED TO LOCATE THE C-HYDROPHONE AND THE
C   ARRAY CENTER.
      DO 12 J = 1,3
         B(4,J) = 0.0D0
         B(5,J) = 0.5*(B(J,1) + B(J,2) + B(J,3))
  12  CONTINUE

      A1 = 0.0D0


C   DETERMINE THE DEPTHS OF THE FOUR HYDROPHONES AND THE ARRAY CENTER.
      HD(1) = A2 + D*(B(5,3) - B(1,3))
      HD(2) = A2 + D*(B(5,3) - B(2,3))
      HD(3) = A2 + D*(B(5,3) - B(3,3))
      HD(4) = A2 + D*(B(5,3) - B(4,3))
      HD(5) = A2

C   FIND THE DEEPEST HYDROPHONE
      A2M = 0.D0
      DO 51 J=1,4
       IF(HD(J).GT.A2M) A2M = HD(J)
  51  CONTINUE

C   LOCATE THE WATER LAYER, N, CONTAINING THE ARRAY.
      N = M
      DO 37 J = 2,M
        IF((LM(J-1).LE.HD(4)).AND.(LM(J).GT.HD(4))) N = J-1
  37  CONTINUE
      IF(N.GE.M)THEN
        PRINT*,'C HYDROPHONE IS DEEPER THAN CURRENT DEPTH-VELOCITY'
        PRINT*,'PROFILE. PROGRAM TERMINATED IN SHIFT SUBROUTINE'
        PRINT*,'POSLOC NUMBER',K
        STOP
      ENDIF
C
```

```
      V = V0(N) + V1(N)*HD(4)

C   CALCULATE THE PRE-TILT CORRECTED APPARENT POSITION
C          WRITE(44,221)K
         DO 40 J = 1,3
           X0(J)=(D**2 + (V*T(K,4)-V*T(K,J))*(V*T(K,4)+V*T(K,J)))/(2*D)
C          WRITE(44,222) J,X0(J)
  40     CONTINUE

C   COMPUTE DIRECTION COSINES .
         CX0 = X0(1)/(V*T(K,4))
         CY0 = X0(2)/(V*T(K,4))
         CZ0 = DSQRT(1.0 - CX0**2 - CY0**2)
C        WRITE(44,223)CX0,CY0,CZ0

C   PERFORM EXACT TILT CORRECTIONS AND THE ROTATIONAL ALIGNMENT.
  42     CX = B(1,1)*CX0 + B(2,1)*CY0 + B(3,1)*CZ0
         CY = B(1,2)*CX0 + B(2,2)*CY0 + B(3,2)*CZ0
         CZ = B(1,3)*CX0 + B(2,3)*CY0 + B(3,3)*CZ0
C        WRITE(44,224)CX,CY,CZ
         SAZ = CY/DSQRT(CX**2 + CY**2)
         CAZ = CX/DSQRT(CX**2 + CY**2)
C        WRITE(44,225)SAZ,CAZ
         PH1 = DATAN2(SAZ,CAZ)
         THONE = 0.5*PIE - DACOS(CZ)
C        WRITE(44,226)PH1,THONE

  49     CALL ISOGAD1(A1,HD(4),T(K,4),THONE,N,LM,VEL,V0,V1,DZ,H2,
     *                 Z2,THE)

         NEW(1) = H2*CAZ
         NEW(2) = H2*SAZ
         NEW(3) = Z2

      RETURN
  221 FORMAT(5X,'POSLOC NUMBER  ',I2/)
  222 FORMAT(10X,'X0',I1,F11.4)
  223 FORMAT(/10X,'CX0 ',F11.4,'  CY0 ',F11.4,' CZ0 ',F11.4)
  224 FORMAT(10X,'CX  ',F11.4,'  CY  ',F11.4,' CZ  ',F11.4)
  225 FORMAT(/10X,'SAZ ',F11.4,'  CAZ ',F11.4)
  226 FORMAT(/10X,'FM SHIFT PH ',F11.4,'  TH ',F11.4)
      END
```

```
***********************************************************************
      SUBROUTINE LONG(TIME,LINKS,SENSR,CPH,K,XS,HD,THONE,TH,PHS,PHL,HO,
     &           XL)
C----------------------------------------------------------------------
C                                                          12/19/90
C   LONGBASE.FOR IS A LONG BASE LINE METHOD FOR TRIPLE CROSSOVER REGIONS
C   THIS IS A MODIFIED VERSION OF LONGBASE FROM REF 3.  WHEN DETERMINING
C   THE ARRAY TO UPDATE (DH) A TOLERANCE LIMIT WAS PLACED ON THE CLOSEST
C   POSLOC DISTANCE SUCH THAT IF WITHIN EPS CHOOSE ARRAY 1 (INTERIOR) TO
C   UPDATE.
C                                                  J.A. GEMBARSKI
C                                                  NPS   01/01/92
C-----------------------------------------------------------------------
C  A1,A2,A3 are the location of the C-Hydrophones.
C  A1(I) is the downrange location of the 3 arrays.
C  A2(I) is the crossrange location of the 3 arrays.
C  A3(I) is the depth location of the 3 arrays.
C        ( Depth is positive down orientation ).
C   K    IS THE OVERLAP REGION OF INTEREST
C  THONE IS THE INITIAL ELEVATION ANGLES TO THE SBL POSLOCS
C   HD   IS THE HORIZONTAL DISTANCE TO THE SBL POSLOCS
C OUTPUTS
C   TH   ELEVATION ANGLES TO THE LONGBASE POSLOCS
C  PHS/PHL  AZIMUTHS TO THE SBL/LBL POSLOCS
C   HO   HORIZONTAL RANGE TO THE LBL POSLOCS
C   XL   LBL POSLOC COORDINATES
***********************************************************************

      DIMENSION A1(3),A2(3),A3(3),X(3),Y(3),Z(3),THO(3),T(3)
      DIMENSION H(3),AR1(2),AR2(2),HD(18),PHO(3),PH1(3)
      DIMENSION XX(3),YY(3),AR(4),LM(300),DZ(300),V1(300)
      DIMENSION L(300),VEL(300),V0(300),XL(6,3),THONE(18),HO(18)
      DIMENSION PHS(18),PHL(18),TH(18),CPH(7,3),XS(18,3)
      DIMENSION LINKS(18),SENSR(7),TIME(18,4)

      COMMON /SET1/ L,VEL,V0,V1,DZ,LM,M

      DOUBLE PRECISION L,VEL,V0,V1,DZ,LM,HO,PHS,PHL,TH
      DOUBLE PRECISION A1,A2,A3,X,Y,Z,DZ0,THO,T,Z0,Z00,H
      DOUBLE PRECISION D12,D13,D23,AR1,AR2,D1,D2,D3,HD,PHO,PH1,XL
      DOUBLE PRECISION VA,VB,UA,UB,EPS,XX,YY,DS,DM,GAM,THONE,TIME,CPH,XS

      INTEGER   DL1,DL2,DH,MARR,AR,ERR,K,LINKS,SENSR,I,M,N

      EPS = 1D-3
      ERR = 0

C  GAM is a convergence tuning constant.
      GAM = 5.0D0

      N = (3*K)-3
      DO 10 I =1,3
         A1(I) = CPH(LINKS(I+N),1)
         A2(I) = CPH(LINKS(I+N),2)
         A3(I) = CPH(LINKS(I+N),3)
         X(I)  = XS(I+N,1)
         Y(I)  = XS(I+N,2)
         Z(I)  = XS(I+N,3)
         H(I)  = HD(I+N)
         THO(I) = THONE(I+N)
         T(I) = TIME(I+N,4)
         AR(I) = SENSR(LINKS(I+N))
   10 CONTINUE


C  Compute the max dz.
      DZO = MAX(DABS(Z(1)-Z(2)),DABS(Z(1)-Z(3)),DABS(Z(2)-Z(3)))

      DO 70 I = 1,3
         PHO(I) = DATAN2(Y(I)-A2(I),X(I)-A1(I))
  70  CONTINUE

C     WRITE(3,280)AR(1),AR(2),AR(3),AR(4)
C     WRITE(3,250)A1(1),A2(1),A3(1)
C     WRITE(3,251)A1(2),A2(2),A3(2)
C     WRITE(3,252)A1(3),A2(3),A3(3)
C     WRITE(3,270)X(1),Y(1),Z(1)
C     WRITE(3,271)X(2),Y(2),Z(2)
C     WRITE(3,272)X(3),Y(3),Z(3)
C     WRITE(3,*)'  '
C     WRITE(3,253)H(1),H(2),H(3)

      D12 = DSQRT((X(2)-A1(1))**2 + (Y(2)-A2(1))**2)
      D13 = DSQRT((X(3)-A1(1))**2 + (Y(3)-A2(1))**2)
      IF ((D12.GE.H(1)).OR.(D13.GE.H(1))) THEN
         ZO = MAX(Z(1),Z(2),Z(3))
      ELSE
         ZO = MIN(Z(1),Z(2),Z(3))
      ENDIF
      DO 75 I = 1,3
   75   IF(Z(I).EQ.ZO) MARR = I
C     WRITE(3,254)DS,ZO

      GOTO 720

   80 D1 = DSQRT((UA-A1(DH))**2 + (VA-A2(DH))**2)
      D2 = DSQRT((UB-A1(DH))**2 + (VB-A2(DH))**2)
      D3 = MIN(D1,D2)
C     WRITE(3,287) D1,D2

      IF(DABS(H(DH) - D3).LT.EPS) GOTO 100

      ZOO = ZO
C     RDH = DSQRT(H(DH)**2 + (A3(DH)-ZO)**2)
      ZO = ZOO + (A3(DH) - ZOO)*(1 - H(DH)/D3)*GAM
      DS = D3 - H(DH)

C     WRITE(3,254)DS,ZO
C     WRITE(3,253)H(1),H(2),H(3)
  720 CONTINUE
      DO 90 I = 1,3
         IF (I.EQ.MARR) GOTO 90
         CALL NEWLOC(0.0D0,A3(I),THO(I),T(I),H(I),Z(I),ZO)
   90 CONTINUE

      IF(MARR.EQ.4) GOTO 98

C  Adjust POSLOCs for new horizontal ranges.
      DO 95 I = 1,3
         IF(I.EQ.MARR) THEN
            XX(I) = X(I)
            YY(I) = Y(I)
         ELSE
            XX(I) = A1(I) + H(I)*DCOS(PHO(I))
            YY(I) = A2(I) + H(I)*DSIN(PHO(I))
         ENDIF
  95  CONTINUE

C  Find the two closest POSLOCs.
      D12 = DSQRT((XX(1) - XX(2))**2 + (YY(1) - YY(2))**2)
      D13 = DSQRT((XX(1) - XX(3))**2 + (YY(1) - YY(3))**2)
      D23 = DSQRT((XX(2) - XX(3))**2 + (YY(2) - YY(3))**2)
      DM = MIN(D12,D13,D23)
C     WRITE(3,288)D12,D13,D23
      IF(DM.LT.EPS) DM = D23
      IF(DM.EQ.D23) THEN
         DL1 = 2
         DL2 = 3
         DH = 1
      ELSEIF(DM.EQ.D13) THEN
         DL1 = 1
         DL2 = 3
         DH = 2
      ELSE
         DL1 = 1
         DL2 = 2
         DH = 3
      ENDIF
C     WRITE(3,255)DH,GAM

      MARR = 4
      AR1(1) = A1(DL1)
      AR2(1) = A2(DL1)
      AR1(2) = A1(DL2)
      AR2(2) = A2(DL2)

  98  CALL CIRCSOLV(AR1,AR2,H(DL1),H(DL2),UA,VA,UB,VB)
C     WRITE(3,285)UA,VA
C     WRITE(3,286)UB,VB
      GOTO 80

  100 CONTINUE

      IF(D3.EQ.D2) THEN
         VA = VB
         UA = UB
      ENDIF
C     WRITE(3,260)UA,VA,ZO
         XL(K,1) = UA
         XL(K,2) = VA
         XL(K,3) = ZO

      DO 110 I = 1,3
         HO(I+N) = H(I)
         PHO(I) = DATAN2(Y(I)-A2(I),X(I)-A1(I))
         PH1(I) = DATAN2(VA-A2(I),UA-A1(I))
         TH(I+N) = THO(I)
         PHS(I+N) = PHO(I)
         PHL(I+N) = PH1(I)
C        WRITE(3,261)AR(I),PHO(I),PH1(I)
C        WRITE(3,262)THONE(I+N),THO(I)
  110 CONTINUE
```

```
      RETURN
 250  FORMAT(/10X,'A1(1)=',F10.2,'   A2(1)=',F10.2,'   A3(1)=',F10.2)
 251  FORMAT(10X,'A1(2)=',F10.2,'   A2(2)=',F10.2,'   A3(2)=',F10.2)
 252  FORMAT(10X,'A1(3)=',F10.2,'   A2(3)=',F10.2,'   A3(3)=',F10.2)
 253  FORMAT(10X,'H(1)=',F11.4,'   H(2)=',F11.4,'   H(3)=',F11.4)
 254  FORMAT(/10X,'DS  =',F11.4,'   ZO  =',F11.4)
 255  FORMAT(/10X,'DH = ',I2,'   GAM = ',F4.1)
 260  FORMAT(//10X,'U   =',F10.2,'   V =',F10.2,'   ZO =',
     *        F10.2)
 261  FORMAT(/10X,'ARR #',I2,'    PHO  = ',F10.6,'    PH1 = ',F10.6)
 262  FORMAT(21X,'THOO = ',F10.6,'    THO = ',F10.6)
 270  FORMAT(/10X,'X(1) =',F10.2,'   Y(1) =',F10.2,'   Z(1) =',F10.2)
 271  FORMAT(10X,'X(2) =',F10.2,'   Y(2) =',F10.2,'   Z(2) =',F10.2)
 272  FORMAT(10X,'X(3) =',F10.2,'   Y(3) =',F10.2,'   Z(3) =',F10.2)
 280  FORMAT(10X,'ARRAYS ',I2,', ',I2,', and ',I2,7X,'CASE ',I2)
 285  FORMAT(/10X,'UA = ',F10.2,'   VA =',F10.2)
 286  FORMAT(10X,'UB = ',F10.2,'   VB =',F10.2)
 287  FORMAT(/10X,'D1 = ',F11.4,'   D2 = ',F11.4)
 288  FORMAT(/10X,'D12 = ',F11.6,'   D13 = ',F11.6,'   D23 = ',F11.6)
      END
C*******************************************************************
      SUBROUTINE NEWLOC(A1,A2,THO,T,HO,Z,ZO)
C------------------------------------------------------------------
C
C  INPUTS:
C     A1:   HORIZONTAL COORDINATE OF SENSOR
C     A2:   VERTICAL COORDINATE OF SENSOR, POSITIVE DOWN
C     M:    INDEX OF DEEPEST LAYER USED
C     THO:  ELEVATION ANGLE
C     T:    TRANSIT TIME
C     HO:   HORIZONTAL RANGE
C     Z:    DEPTH OF POSLOC
C     ZO:   DEPTH GOAL
C
C OUTPUTS:
C     THP:  ADJUSTED ELEVATION ANGLE
C
C------------------------------------------------------------------
      DIMENSION L(300),VEL(300),VO(300),V1(300),DZ(300),LM(300)
      COMMON /SET1/ L,VEL,VO,V1,DZ,LM,M
      DOUBLE PRECISION L,VEL,VO,V1,DZ,LM
      DOUBLE PRECISION THO,T,HO,Z,ZO,THP,EPS,A1,A2,TH1
      INTEGER J,M,N

C  compute the slant range
      EPS = .0010D0
      DO 5 J = 2,M
        IF((LM(J-1).LE.A2).AND.(LM(J).GT.A2)) N = J-1
*       PRINT*,'N',N,'A2',A2,'LMJ-1',LM(J-1),'LMJ',LM(J)
   5  CONTINUE

      THP = THO
  10  CONTINUE
      THP = THP - (ZO-Z)/HO
*     PRINT*,'N',N
      CALL ISOGAD1(A1,A2,T,THP,N,LM,VEL,VO,V1,DZ,HO,Z,TH1)
      IF (DABS(Z-ZO).GE.EPS) GOTO 10

      THO = THP
      RETURN
      END

C******************************************************************
      SUBROUTINE CIRCSOLV(A1,A2,H1,H2,U1,V1,U2,V2)
C------------------------------------------------------------------
C  INPUTS:
C     A1: CENTER OF THE 1ST CIRCLE
C     A2: CENTER OF THE 2ND CIRCLE
C     H1,H2:  RADIUS
C  OUTPUTS:
C     U1,V1:  1ST INTERSECTION POINT
C     U2,V2:  2ND INTERSECTION POINT
C------------------------------------------------------------------
      DIMENSION A1(2),A2(2)
      DOUBLE PRECISION A1,A2,B0,B1,H1,H2,U1,U2,V1,V2,P1,P2,P3,B2
      INTEGER   ERR

      ERR = 0
      B0 = H1**2 - H2**2 + A1(2)**2 - A1(1)**2 + A2(2)**2 - A2(1)**2
      B1 = 2*(A1(2) - A1(1))
      B2 = 2*(A2(2) - A2(1))
      B1 = -B1/B2
      B0 = B0/B2
      P1 = 1+B1**2
      P2 = 2*B1*(B0-A2(1))-2*A1(1)
      P3 = A1(1)**2 + (B0-A2(1))**2 - H1**2
      CALL QUAD(P1,P2,P3,U1,U2,ERR)
      IF (ERR.EQ.1) THEN
          WRITE(*,*)'  THERE WAS AN ERROR IN QUAD SOLUTION'
          STOP
      ENDIF
      V1 = B0 + B1*U1
      V2 = B0 + B1*U2
      RETURN
      END

C*********************************************************************
      SUBROUTINE QUAD(A,B,C,X1,X2,ERR)
C---------------------------------------------------------------------

      DOUBLE PRECISION A,B,C,D,X1,X2
      INTEGER   ERR

      D = B**2 - 4.0*A*C
      IF (D.LT.0.0D0) THEN
         ERR = 1
         GOTO 10
      ENDIF
      X1 = (-B + DSQRT(D))/(2.0*A)
      X2 = (-B - DSQRT(D))/(2.0*A)
   10 RETURN
      END
```

APPENDIX G: COMPUTER PROGRAM VELPRO

The VELPRO program sets up the water column for use in RAYFIT and ISOGAD subroutines. It also performs any DVP extrapolation required if an array hydrophone should exceed the deepest depth of the current profile. The extrapolation control can be overridden from the calling program.

```
      SUBROUTINE VELPRO(OP,NAME1,N,EXT,A2)
*
*   THIS A COMBINATION OF FORTRAN PROGRAMS VELMOD (2/22/90) AND
*   ERRPROP (8/22/90) IN LIB12.FOR OF REF 1.
*  - A CORRECTION WAS MADE TO THE EXTRAPOLATION PROCEDURE FROM ERRPROP
*    IN THE TERM GG.  THE FACTOR OF 21 WAS INADVERTENTLY LEFT OUT OF
*    THE DENOMINATOR IN THE ORIGINAL PROGRAM.
*   THIS SUBROUTINE READS IN THE DEPTH VELOCITY PROFILE AND THEN
*   PERFORMS A DEPTH EXTENSION IF A2 IS DEEPER THAN THE LAST LAYER
*                                          J.A. GEMBARSKI
*                                          NPS   01/01/92
*----------------------------------------------------------------------
      DIMENSION L(300),VEL(300),V0(300),V1(300),DZ(300),LM(300)

      COMMON /SET1/ L,VEL,V0,V1,DZ,LM,M

      DOUBLE PRECISION    L,VEL,LM,DZ,V0,V1,K0,G,GG,SV,SVU2,SVU,SU2,SU4
      DOUBLE PRECISION    U,LB,A2
      INTEGER   EXT,M,OP,N
      CHARACTER*15 NAME1

      IF (EXT.EQ.1) GOTO  111

C  READ IN THE DEPTH AND VELOCITY ARRAYS
      IF(N.EQ.1) NAME1 = '/ON880512 DVT A'
      IF(N.EQ.2) NAME1 = '/ON880622 DVT A'
      IF(N.EQ.3) NAME1 = '/ON880721 DVT A'
      IF(N.EQ.4) NAME1 = '/ON880803 DVT A'
      IF(N.EQ.5) NAME1 = '/ON881027 DVT A'
      IF(N.EQ.6) NAME1 = '/ON890113 DVT A'
      IF(N.EQ.7) NAME1 = '/ON890308 DVT A'
      IF(N.EQ.8) NAME1 = '/ON890322 DVT A'
      IF(N.EQ.9) NAME1 = '/ON890426 DVT A'
      IF(N.EQ.10) NAME1 = '/ONTEST0  DVT A'
      IF(N.EQ.11) NAME1 = '/ONTEST2  DVT A'
      IF(N.EQ.12) NAME1 = '/ONTEST4  DVT A'
      IF(N.EQ.13) NAME1 = '/ONTEST6  DVT A'
      IF(N.EQ.14) NAME1 = '/ONTEST8  DVT A'
      IF(N.EQ.15) NAME1 = '/ONTEST83 DVT A'
      IF(OP.EQ.1)PRINT*,  'LOAD VELOCITY FILE: ',NAME1
C     IF(OP.EQ.1)WRITE(91,*)'LOAD VELOCITY FILE: ',NAME1
      IF(OP.EQ.1)WRITE(92,*)'LOAD VELOCITY FILE: ',NAME1
      OPEN(UNIT=4,FILE=NAME1,STATUS='OLD')
      I = 1
   5  CONTINUE
```

```
      READ(4,130,ERR=7,END=10)L(I),VEL(I)
         L(I) = -L(I)
         I = I+1
      GOTO 5
   7  WRITE(*,*)'      ! THERE WAS AN ERROR READING THE FILE !'
      STOP
  10  CONTINUE
      CLOSE(UNIT=4)
      M = I-1
*
*                FORM THE SET OF LAYER MIDPOINTS
*
      DO 40 I=1,M-1
         LM(I) = .5*(L(I) + L(I+1))
   40 CONTINUE
*
* FORM DEPTH INCREMENTS, AND ALL SOUND VELOCITY SLOPES AND INTERCEPTS
*
      DO 50 I=1,M-2
         DZ(I) = LM(I+1) - LM(I)
         V0(I) = (LM(I+1)*VEL(I) - LM(I)*VEL(I+1))/DZ(I)
         V1(I) = (VEL(I+1) - VEL(I))/DZ(I)
   50 CONTINUE
         LM(M) = LM(M-1) + DZ(M-2)
 130  FORMAT(5X,D8.2,5X,D7.2)

111   IF(A2.LT.LM(M-1) .OR. EXT.EQ.2) GOTO 126

C   EXTRAPOLATE
C   THE SOUND VELOCITY PROFILE BY USING A QUADRATIC FUNCTION OVER
C   THE DEEPEST 100 FEET.

C   FIRST COUNT THE NUMBER OF LAYERS (OF THICKNESS DZ(M-2)) TO
C   BE ADJOINED.  ALSO MUST EXTEND THE L ARRAY.

      K0 = 2 + MAX(0,NINT((A2-LM(M-1))/DZ(M-2)))

C   FIND AVERAGE DEPTH OF LAST 100 FEET.
      LB = 0.0D0
      DO 43 I = M-21,M-1
         LB = LB + LM(I)
  43  CONTINUE
      LB = LB/21

C   FORM SUMS OF POWERS AND PRODUCTS.
      SV = 0.0D0
      SVU2 = 0.0D0
      SVU = 0.0D0
      SU2 = 0.0D0
      SU4 = 0.0D0
      DO 45 I = M-21,M-1
         U = LM(I) - LB
         SV = SV + VEL(I)
         SVU = SVU + U*VEL(I)
         SVU2 = SVU2 + U**2 * VEL(I)
         SU2 = SU2 + U**2
         SU4 = SU4 + U**4
  45  CONTINUE

      G = SVU/SU2
      GG = (21*SVU2 - SU2*SV)/(21*SU4 - SU2**2)
      V1(M-1) = G

C   PERFORM THE EXTRAPOLATION.
      DO 125 I=M,M+KO
         V1(I-1) = V1(I-2) + GG*DZ(M-2)
         LM(I) = LM(I-1) + DZ(M-2)
         VEL(I) = VEL(I-1) + DZ(M-2)*V1(I-1)
         VO(I-1) = (LM(I)*VEL(I-1) - LM(I-1)*VEL(I))/DZ(M-2)
         L(I+1) = L(I) + DZ(M-2)
         DZ(I-1) = DZ(M-2)
  125 CONTINUE

C  UPDATE M, THE NUMBER OF LAYERS

      M = M + KO
  126 CONTINUE
      RETURN
      END
```

APPENDIX H: REGIONAL POSLOC DATA (INTERIOR ARRAY 15) FOR RANGE CORRECTION

REGIONAL POSLOC DATA INTERIOR ARRAY 15
SATELLITE ARRAYS WITH ERRORS - ISOSPEED

REGION 13

	ARRAY 15 SBL	ARRAY 14 SBL	ARRAY 24 SBL	LONGBASE	GOAL
X	49520.90	49487.10	49482.90	49510.14	49493.30
Y	-8605.20	-8607.33	-8590.08	-8588.79	-8593.79
Z	425.14	375.98	424.18	428.80	400.00
THS	0.20431	0.21783	0.20157		
THL	0.20347	0.20578	0.20051		
PHS	-2.59788	-0.50234	1.58702		
PHL	-2.60237	-0.49611	1.58078		
HS	4364.28	4376.90	4367.09		
HL	4365.04	4388.25	4368.03		

REGION 14

	ARRAY 15 SBL	ARRAY 5 SBL	ARRAY 14 SBL	LONGBASE	GOAL
X	49390.02	49412.46	49407.20	49401.91	49404.92
Y	-4369.35	-4360.42	-4370.22	-4360.26	-4356.84
Z	417.28	386.64	385.75	387.20	400.00
THS	0.20704	0.21490	0.21855		
THL	0.21397	0.21477	0.21821		
PHS	2.66863	-1.59600	0.51579		
PHL	2.66552	-1.59845	0.51840		
HS	4342.51	4317.04	4318.03		
HL	4336.08	4317.16	4318.35		

REGION 15

	ARRAY 15 SBL	ARRAY 5 SBL	ARRAY 6 SBL	LONGBASE	GOAL
X	53197.79	53245.52	53236.15	53241.45	53235.75
Y	-2059.94	-2093.16	-2091.73	-2069.93	-2085.69
Z	389.46	408.91	411.46	343.39	400.00
THS	0.21580	0.21317	0.21216		
THL	0.22656	0.22861	0.22820		
PHS	1.58433	-0.50286	-2.63156		
PHL	1.57415	-0.49851	-2.63544		
HS	4287.92	4250.41	4250.74		
HL	4277.56	4235.70	4235.51		

REGION 21

	ARRAY 15 SBL	ARRAY 6 SBL	ARRAY 16 SBL	LONGBASE	GOAL
X	57039.28	57062.13	57062.38	57071.57	57054.98
Y	-4266.55	-4325.19	-4325.69	-4319.17	-4330.68
Z	370.65	412.73	409.32	385.99	400.00
THS	0.21850	0.20903	0.21008		
THL	0.21495	0.21524	0.21547		
PHS	0.50285	-1.54383	2.65266		
PHL	0.48858	-1.54160	2.65032		
HS	4317.97	4310.26	4330.69		
HL	4321.34	4304.50	4325.65		

REGION 22

	ARRAY 15 SBL	ARRAY 16 SBL	ARRAY 25 SBL	LONGBASE	GOAL
X	57081.17	57057.44	57056.64	57053.99	57054.15
Y	-8523.61	-8582.94	-8588.56	-8586.27	-8587.96
Z	379.08	394.12	404.19	416.36	400.00
THS	0.21267	0.20893	0.18296		
THL	0.20421	0.20391	0.18021		
PHS	-0.51722	-2.61546	1.55218		
PHL	-0.53262	-2.61520	1.55279		
HS	4401.01	4426.93	4416.72		
HL	4408.89	4431.59	4418.96		

REGION 23

	ARRAY 15 SBL	ARRAY 24 SBL	ARRAY 25 SBL	LONGBASE	GOAL
X	53341.65	53288.99	53293.50	53294.50	53293.11
Y	-10680.56	-10692.31	-10704.16	-10690.85	-10707.34
Z	406.18	425.32	418.14	452.52	400.00
THS	0.20989	0.20128	0.18299		
THL	0.19921	0.19505	0.17508		
PHS	-1.55099	0.54496	2.58304		
PHL	-1.56189	0.54459	2.58032		
HS	4333.94	4367.96	4340.60		
HL	4343.55	4373.42	4346.82		

APPENDIX I: FORMULATION OF THE PROBLEM AS A NONLINEAR PROGRAM AND ALGORITHM CHOICE

Using the objective function defined in Ch. III, the mathematical formulation for the array locational correction problem is stated as follows:

INDICES:

i coordinate index
j triple overlap region index, $j=1,\dots,NR_k$
k sensor array index

GIVEN DATA:

$\{X^o_{k,i}\}$ initial position (x,y,z) for array k for i=1,2,3
$\{A^o_{k,i}\}$ initial orientation (xtilt,ytilt,zrot) for array k for i=1,2,3
NR_k number of triple overlap regions for array k
v(z) the DVP of the water as a function of depth
t the transit times of the acoustic pings

VARIABLES:

Decision:
$SHIFT_{k,i}$ = location shift (x,y,z,xtilt,ytilt,zrot) for array k for i=1,...,6

Internal:
$\{ZER_{k,j}\}$ = depth error (z) for array k in region j
$\{HER_{k,j}\}$ = horizontal range error for array k in region j
$\{PHER_{k,j}\}$ = azimuth angle error for array k in region j
$\{XS_{k,j,i}\}$= SBL posloc (x,y,z) from array k in region j for i=1,2,3, this is a function the arrays location (decision variables), the DVP and transit time, evaluated through the ray tracing algorithms
$\{XL_{k,j,i}\}$= LBL posloc (x,y,z) from array k in region j for i=1,2,3, this is a similar function to the above, but uses three independent arrays

$\{HL_{k,j}\}$ = horizontal range from array k to LBL posloc in region j
$\{HS_{k,j}\}$ = horizontal range from array k to SBL posloc in region j
$\{PHL_{k,j}\}$ = azimuth angle from array k to LBL posloc in region j
$\{PHS_{k,j}\}$ = azimuth angle from array k to SBL posloc in region j

CONSTRAINTS:

$$HL_{k,j} = \sqrt{XL^2_{k,j,1} + XL^2_{k,j,2}} \qquad \textit{for all } k,j$$

$$HS_{k,j} = \sqrt{XS^2_{k,j,1} + XS^2_{k,j,2}} \qquad \textit{for all } k,j$$

f(v(z),t) = this function is used to construct the acoustic ray path

$XS_{k,j,i}$ = $f_1(X_{k,i}, A_{k,i};\ f(v(z),t))$
for all k,j i=1,2,3
(this function is evaluated using the iterative ray tracing algorithms [Ref. 1])

$XL_{k,j,i}$ = $f_2((X_{k,i}, A_{k,i})_j;\ f(v(z),t))$
for all k,j i=1,2,3
(this function is similar to the above (f_1), but uses three independent arrays, for region j, in the posloc evaluation [Ref. 3])

$HER_{k,j}$ = $HL_{k,j} - HS_{k,j}$ for all k,j

$ZER_{k,j}$ = $XL_{k,j,3} - XS_{k,j,3}$ for all k,j

$PHL_{k,j}$ = $\arctan[(XL_{k,j,2}-X_{k,2})/(XL_{k,j,1}-X_{k,1})]$
for all k,j i=1,2,3

$PHS_{k,j}$ = $\arctan[(XS_{k,j,2}-X_{k,2})/(XS_{k,j,1}-X_{k,1})]$
for all k,j i=1,2,3

$PHER_{k,j}$ = $PHL_{k,j} - PHS_{k,j}$ for all k,j

$X_{k,i}$ = $X^o_{k,i} + SHIFT_{k,i}$ for all k, i=1,2,3
$A_{k,i}$ = $A^o_{k,i} + SHIFT_{k,(i+3)}$ for all k, i=1,2,3

$$-\pi/2 \le A_{k,i} \le \pi/2 \quad \textit{for all k, i=1,2}$$

$$-\pi \le A_{k,3} \le \pi \quad \textit{for all k}$$

$X_{k,i}$ within the RANGE coordinate system for all k, i=1,2,3

$X_{k,3}$ > 0 (array must be below water) for all k

OBJECTIVE FUNCTION:

$$MIN \sum_{k} \sum_{j=1}^{NR_k} \sqrt{ZER_{k,j}^2 + HER_{k,j}^2 + (HL_{k,j} \times PHER_{k,j})^2}$$

for all k with NR_k = 6

The first summation in the objective function is over the set of index k which yields the value NR_k =6. This isolates the problem to the interior arrays only.

Due to the nature of underwater ray tracing, the objective function and constraints are nonconvex and nonlinear. This environmental nature also prevents the analytical expression of the functions f_1 and f_2 (see the dynamic programming solution outlined in Ref 1). Without the analytical expressions of these functions the gradients are also unavailable in analytical form. Based on the above, any algorithm requiring the evaluation of the gradient would be too time consuming requiring the evaluation of the functions to obtain the gradients. With this in mind, we choose an algorithm that does not require the evaluation of a gradient. Among the algorithms available without using a gradient are the cyclic coordinate descent algorithms. These algorithms allow the problem structure to be more efficiently exploited.

A simulated annealing algorithm was used to determine the (global) optimality of the solution provided by ERRFIX. This algorithm allowed alternate local solutions to the problem to be found. Based on computational experiments, solutions from simulated annealing were only marginally better than those obtained using ERRFIX. Thus it is hypothesized that ERRFIX provides a nearly global optimal solution. Letting ERRFIX perform more iterations may enhance the quality of the solution produced.

LIST OF REFERENCES

1. Read, Robert R., A Study of Underwater Sound Ray Tracing Methodology, NPS Technical Report, NPS55-90-21, September, 1990.

2. Read, Robert R., An Investigation of Timing Synchronization Errors for Tracking Underwater Vehicles, NPS Technical Report, NPS55-90-15, July, 1990.

3. Read, Robert R., A Technique foe Assessing Short Baseline Array Tilt Errors, NPS Technical Report, NPSOR-91-13, May, 1991.

4. Read, Robert R., Program for the Simultaneous Estimation of Displacement and Orientation Corrections for Several Short Baseline Arrays, NPS Technical Report, NPS55-85-028, November, 1985.

5. Gygax, Gene, The Simulation of Remotely Measured Paths of Underwater Vehicles for the Purpose of Monitoring the Calibration of Test Ranges, Master of Science Thesis, Naval Postgraduate School, Monterey, California, September, 1985.

6. Kroshl, William M., Methodologies for Resolving Anomalous Position Information in Torpedo Range Tracking Using Simulation, Master of Science Thesis, Naval Postgraduate School, Monterey, California, March, 1988.

7. Camp, Leon, Underwater Acoustics, John Wiley and Sons, New York, N.Y., 1970.

8. Bohachevsky, Ihor, O. et. al., Generalized Simulated Annealing for Function Optimization, Technometrics, Vol. 28, No. 3, August 1986.

9. IMSL, Problem Solving Software Systems, 1985.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   Cameron Station
   Alexandria, Virginia 22304-6145

2. Library, Code 52 2
   Naval Postgraduate School
   Monterey, California 93943-5002

3. Robert R. Read, Code OR-Re 3
   Operations Research Department
   Naval Postgraduate School
   Monterey, California 93943-5002

4. Nava' Undersea Warfare Engineering Station 2
   Keyport, Washington 98345
   ATTN: J. Knudsen, Code 5122

5. CMSgt. K. S. Gembarski USAF (Ret) 3
   5230 Cedar Lake Road
   Oscoda, Michigan 48750